Technology in conservation: towards a system for in-field drone detection of invasive vegetation

Submitted in fulfilment
of the requirements for the degree of

Master of Science

of Rhodes University

Katherine Margaret Frances James

https://orcid.org/0000-0003-0901-3791

Grahamstown, South Africa
December 27, 2019

Abstract

Remote sensing can assist in monitoring the spread of invasive vegetation. The adoption of camera-carrying unmanned aerial vehicles, commonly referred to as drones, as remote sensing tools has yielded images of higher spatial resolution than traditional techniques. Drones also have the potential to interact with the environment through the delivery of bio-control or herbicide, as seen with their adoption in precision agriculture. Unlike in agricultural applications, however, invasive plants do not have a predictable position relative to each other within the environment. To facilitate the adoption of drones as an environmental monitoring and management tool, drones need to be able to intelligently distinguish between invasive and non-invasive vegetation on the fly.

In this thesis, we present the augmentation of a commercially available drone with a deep machine learning model to investigate the viability of differentiating between an invasive shrub and other vegetation. As a case study, this was applied to the shrub genus *Hakea*, originating in Australia and invasive in several countries including South Africa. However, for this research, the methodology is important, rather than the chosen target plant. A dataset was collected using the available drone and manually annotated to facilitate the supervised training of the model. Two approaches were explored, namely, classification and semantic segmentation. For each of these, several models were trained and evaluated to find the optimal one. The chosen model was then interfaced with the drone via an Android application on a mobile device and its performance was preliminarily evaluated in the field. Based on these findings, refinements were made and thereafter a thorough field evaluation was performed to determine the best conditions for model operation.

Results from the classification task show that deep learning models are capable of distinguishing between target and other shrubs in ideal candidate windows. However, classification in this manner is restricted by the proposal of such candidate windows. End-to-end image segmentation using deep learning overcomes this problem, classifying the image in a pixel-wise manner. Furthermore, the use of appropriate loss functions was found to improve model performance. Field tests show that illumination and shadow pose challenges to the model, but that good recall can be achieved when the conditions are ideal. False positive detection remains an issue that could be improved. This approach shows the potential for drones as an environmental monitoring and management tool when coupled with deep machine learning techniques and outlines potential problems that may be encountered.

ACM Computing Classification System Classification

Thesis classification under the ACM Computing Classification System (2012 version, valid through 2019) [1]:

Computing methodologies
→ **Computer Vision** → Image Segmentation; Vision for robotics.
→ **Machine Learning** → Neural Networks

[1] https://www.acm.org/publications/class-2012

Acknowledgements

At this, the beginning of my thesis and end of this chapter of my life, there are many to whom I am grateful. Foremost among these is my wonderful and dedicated supervisor Karen, for convincing me to do a masters degree in the first place and for all her advice, guidance and support along the way. To my dear family, almost family and good friends for all the love, encouragement and support along the way, I thank you.

Thanks must also go to the Rhodes University Ian Mackenzie and Henderson Scholarship funds for the financial support that has made this work possible, as well as the Telkom Center of Excellence.

This work was undertaken in the Distributed Multimedia COE at Rhodes University, with financial support from Telkom SA, Tellabs, Easttel, Bright Ideas 39, THRIP and NRF SA (UID 75107). The opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.

Contents

List of Figures

List of Tables

Listings

Abbreviations

API	Application programming interface
BCE	Binary cross-entropy
CCI	Counter class imbalance (referring to loss functions)
CE	Cross-entropy
CIVE	Colour index of vegetation extraction
CNN	Convolutional neural network
COM	Combination vegetation index
COM2	Combination vegetation index 2
CPU	Central processing unit
DEM	Digital elevation model
DSM	Digital surface model
DTM	Digital terrain model
DVI	Difference vegetation index
EV	Exposure value
EVI	Enhanced vegetation index
ExG	Excess green
ExGR	Excess green minus excess red, a vegetation index
FCN	Fully convolutional network
FN	False negatives
FP	False positives
FPV	First person view
GLI	Green leaf index
GPU	Graphical processing unit
HOG	Histogram of oriented gradients
HSV	Hue Saturation Value – a colour space
IOU	Intersection over union
KNN	K-nearest neighbours
LiDAR	Light detection and ranging

NDVI	Normalised difference vegetation index
NGRDI	Normalised green-red difference
NIR	Near infrared
OBIA	Object-based image analysis
PBIA	Pixel-based image analysis
R-CNN	Region-based CNN
R-FCN	Region-based fully convolutional network
GCC	Green chromatic coordinate
RGB	Red green blue – a colour space
ROI	Region of interest
SDK	Software development kit
SfM	Structure from motion
SSD	Single shot multibox detector
SVM	Support vector machine
TN	True negatives
TP	True positives
UI	User interface
UX	User experience
VARI	Visible atmospheric resistant index
VEG	Vegetative index
WBCE	Class weighted BCE
WF	Weighted focal
WI	Woebbecke index
YFI	Yellow flag iris
YOLO	You only look once
YUV	A colour space

Chapter 1

Introduction

1.1 Context of the research

According to the Department of Water Affairs (1999), South Africa has 198 invasive plant species that have spread to cover ten million hectares of land and are still spreading rapidly. Invasive alien plants can be the prime cause of the endangerment of indigenous flora (Pimentel *et al.*, 2005), moving them along the extinction trajectory (Downey & Richardson, 2016), which is a major threat to biodiversity. According to Richardson & Van Wilgen (2004), these species also shift the ecosystem balance, altering soil composition, using excessive resources, altering fire regimes and in some cases causing erosion. Additionally, these species are well known for their negative effect on water resources and are estimated to have caused a reduction in water flows of 1444 million $m^3.yr^{-1}$ (Le Maitre *et al.*, 2016). These factors also lead to major related financial losses (Pimentel *et al.*, 2005).

The threat posed by invasive alien plants within South Africa led to the initiation of the Working for Water[1] program in 1995, administered by the Department of Water Affairs and Forestry. This program aims to integrate poverty reduction with environmental conservation, with the clearing of alien invasives through the employment of local communities (Buch & Dixon, 2009). During their time of operation Working for Water has cleared approximately one million hectares of invaded land, using three methods of invasive plant control, namely mechanical, chemical and biological control agents (Department of Water Affairs, 1999).

[1] https://www.environment.gov.za/projectsprogrammes/wfw

Despite the efforts of Working for Water, the country has seen an exponential spread of alien invasives (Department of Water Affairs, 1999). This suggests that additional measures need to be put in place to keep up with the spread of exotic vegetation within the country.

Remote sensing, using satellites and manned flights, has been conducted to study invasive plants, with many applications of this in the reviews of Huang & Asner (2009) and Bradley (2014). Detection of these plants is usually made through spectral, textural of phenological approaches, with the most common being a spectral approach using high-resolution imagery with at least four spectral bands (Bradley, 2014). Rather than species-level detection from remotely sensed data, Bradley contends that it is more common for the detection of functional plant type to occur, used for the creation of land cover maps, classifying areas as shrubland, for example. In the case that species-level detection is performed, Bradley writes that high-resolution imagery must be available and that the species, compared to the surrounding vegetation, possesses a sufficiently unique spectral, textural or phenological trait.

A new source of high-resolution remotely sourced imagery has become available with the advent of affordable commercial Unmanned Aerial Vehicles, commonly referred to as 'drones'. These provide the benefit of a lower associated cost than manned flights, as well as offering control over the temporal resolution of the images (Cruzan *et al.*, 2016). In addition to their use in research tasks, the adoption of drones has seen a rise in the agricultural sector. Drones have been used to detect weeds in various crop fields, such as maize (Peña *et al.*, 2013), sunflowers (Torres-Sánchez *et al.*, 2013), barley (Franco *et al.*, 2018) and wheat (Torres-Sánchez *et al.*, 2014), amongst others. This data can be used to create a weed-map or can be interpreted in real-time for site-specific herbicide application (Franco *et al.*, 2018).

Drones are also proving promising for environmental monitoring, notably in the mapping and monitoring of invasive alien plants. Various case studies have been conducted for the detection of invasive alien plants using drone imagery and several authors have found that using drone imagery for detecting invasive alien plants provides better accuracies than when using satellite imagery (Martin *et al.*, 2018, Müllerová *et al.*, 2017).

The relatively recent success of deep learning methods has begun to see their incorporation into all stages of remote sensing data analysis (Zhang *et al.*, 2016). With regard to plant identification, there are a growing number of studies in which deep learning methods are out-performing traditional detection methods, such as that by Guirado *et al.* (2017).

The continued spread of alien plant invasions contends that new methodology is necessary to assist in the mapping and monitoring of these species. Furthermore, the success of deep learning methods in conjunction with the availability of high-resolution aerial imagery from drones suggests a promising way to detect alien plant invasions. Precision agriculture suggests that if it is possible to produce reliable detections, in the future drones may be used to assist beyond the monitoring of invasive alien plants, potentially assisting in keeping these invasions in check through the delivery of bio-control or site-specific herbicide. However, before any of this is possible, it is necessary to determine to what extent a reliable detection can be made in the field and what the best way to approach this is.

1.2 Research statement and objectives

This project aims to explore the use of technology in conservation, with application to drone detection of alien invasive plants. In particular, this relates to exploring the capacity of drones for detection of invasive plants in the field, rather than using post-collection processing on the data.

The primary objective of this investigation is thus to determine whether a commercially available drone augmented with a deep learning model is able to detect invasive plants in the field. This objective is divided into several sub-objectives, which are as follows:

1. To determine which deep learning approach is best suited to achieve the primary objective.
2. To investigate what optimal level of performance can be achieved in detection.
3. To investigate the suitability of the augmented drone system for detection in the field.

1.3 Approach

A number of tasks were conducted to achieve each of the sub-objectives. Firstly, different deep learning approaches were evaluated, namely, classification and segmentation. Thereafter, a model of the most suitable architecture was refined using different loss functions to obtain a measure of the best attainable performance. Finally, the best performing

model was used to augment the drone and the resulting system was trialled in the field and its performance analysed.

1.4 Thesis organisation

The remainder of this thesis comprises the following chapters:

Background Concepts (Chapter 2)— introduces background concepts of relevant drone technology systems, image processing and machine learning techniques.

Related Work (Chapter 3)— introduces the relevant work from the literature on the use of drones as a research tool and detection techniques in drone sourced aerial imagery.

Research Design and Methodology (Chapter 4)— presents an overview of the approach taken in this thesis.

Model Design (Chapter 5)— discusses the development of a model for detecting the target plant, in which different approaches are contrasted to determine the most optimal one for the task.

Application design and preliminary testing (Chapter 6)— documents the integration of the trained model with the Android application, built with the DJI SDK, for drone control. The results of preliminary testing in the field are also presented.

Model Refinement (Chapter 7)— documents experiments to improve model performance based on findings from preliminary testing, making use of further augmentation and different loss functions.

Field Testing (Chapter 8)— presents the results of testing to determine the limits of operation of the chosen model under varying environmental and image-capture conditions.

Conclusion (Chapter 9)— discusses the thesis findings with concluding remarks, as well as mentioning future work that could be carried out.

Chapter 2

Background Concepts

This chapter introduces background concepts and terminology essential to this research. Four topics are covered, namely unmanned aerial systems, digital image processing, remote sensing and photogrammetric mapping, and machine learning platforms. In each of these sections, relevant definitions and techniques are highlighted.

2.1 Unmanned aerial systems

The term *drone* can refer to both fixed-wing and rotary-wing unmanned aircraft. According to the review by Floreano & Wood (2015), fixed-wing aircraft are more efficient aerodynamically and therefore generally have longer flight-times than rotary winged aircraft. However, these aircraft require either a runway or launcher for take-off as well as constant motion for flight, so they cannot hover. Rotary wing aircraft, on the other hand, such as helicopters, quadcopters and hexacopters, are less aerodynamically efficient than fixed-wing aircraft but are able to hover in place (Floreano & Wood, 2015). They also have the additional benefits of agile manoeuvrability and vertical take-off and landing. In this work, the term *drone* is used to refer to small rotary-wing unmanned aircraft, particularly quadcopters.

Quadcopters have four rotors, two that spin clockwise and two that spin counterclockwise. Varying the speed of these rotors provides thrust, yaw and lift for the quadcopter (Ghazbi *et al.*, 2016). In the past few years the use of quadcopters has grown with a wide range of functionalities, which Colomina & Molina (2014) grouped under the following categories: "Agricultural and environmental applications", "Intelligence, surveillance and

reconnaissance", "Aerial monitoring in engineering", "Cultural heritage" – particularly the surveyance of archaeological sites, and "Traditional surveying, conventional mapping and photogrammetry, and cadastral applications".

According to Colomina & Molina (2014), Unmanned Aerial Systems (UAS) are defined to consist of a drone, ground control system and pilot or autopilot for communication between the two, each of which is discussed in greater detail in the following subsections.

2.1.1 Ground control station

A ground control station or groundstation is a control centre running software that controls, monitors and interacts remotely with a drone (Colomina & Molina, 2014). Operated by either a pilot or an autopilot, common groundstations are often operated from computers or mobile phones.

2.1.2 Autopilot

In their survey of autopilots for small unmanned aerial vehicles, Chao *et al.* (2010) defined an autopilot as a closed-loop system which controls the flight of drones using a combination of hardware and software and without human involvement. They go on to say that autopilots have both a state observer, which contains an inertial measurement unit, and a controller, which issues commands to the drone. Furthermore, the authors note that autopilots can be based on proportional-integral-derivatives, fuzzy logic, or neural networks, amongst others. Floreano & Wood (2015) note that there are three levels of autonomy that can be achieved. These are sensory-motor autonomy, in which high-level commands are executed, reactive autonomy, in which actions are taken to compensate for environmental perturbations, and cognitive autonomy, in which a drone can plan actions, learn, recognise objects and navigate around them.

Mission planning is an important step for autonomous flight (Colomina & Molina, 2014). An example of the functionality available through the mission planning software of the commercially available DJI drones is waypoint navigation – through a set of predefined GPS locations, and functions like return to home, circle point of interest or follow an object of interest (DJI, 2017).

2.1.3 Communication and control

Data links between a drone and its groundstation are usually via WiFi around the 2.4 GHz band, although some military drones use high-frequency satellite communication, as reported by Colomina & Molina (2014). There are generally two important data links between a groundstation and a drone: the downlink from the drone to the groundstation and the uplink from the ground station to the drone (Gupta *et al.*, 2013). The downlink carries telemetry information pertinent to the current status of the drone, while the uplink contains control information relating to updates of the flight plan (Gupta *et al.*, 2013).

2.1.4 Supporting sensors

Drones require sensors both to determine their state and to perform their function using their payload. This subsection gives a brief overview of common sensors used for drone functionality.

Cameras

Cameras are by far the most common sensor payload for commercial drones, allowing for both still photo and video footage collection and are vital when conducting remote sensing surveys used to build representations, models and maps of the environment. A number of different camera types are available, each providing different information. These are visible-band, near-infrared (NIR), multi-spectral and hyper-spectral cameras, which provide a higher spectral resolution than the multi-spectral cameras (Colomina & Molina, 2014). The combined use of visible and NIR bands is important for calculating vegetation indices, such as the well known Difference Vegetation Index (NDVI), a common step in vegetation mapping. Hyper-spectral cameras are occasionally used for imaging vegetation, but not commonly for mapping due to a poor signal-to-noise ratio (Huang & Asner, 2009)

Inertial measurement unit and GPS

GPS measurements combined with sensor information from the inertial measurement unit are important for state estimation (Chao *et al.*, 2010). The inertial measurement

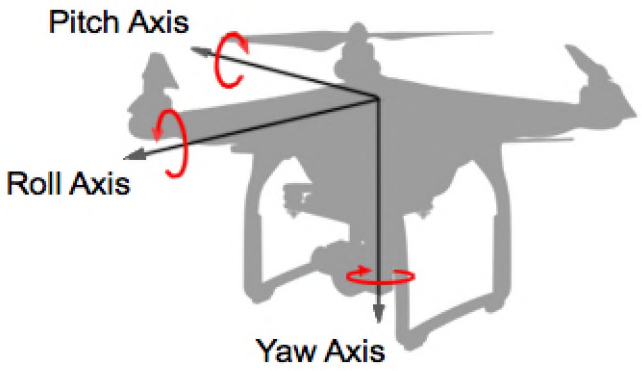


Figure 2.1: Yaw, pitch and roll directions[1].

unit contains three-axis accelerometers, gyroscopes, a magnetometer, and occasionally a barometer and GPS module, which provide readings to be used in the calculation of drone state in terms of yaw, pitch and roll (Floreano & Wood, 2015). These axes are illustrated in Figure 2.1.

GPS modules use four satellites and their known positions to calculate relative distances between the module and the satellites and thus determine the GPS module's position (Vale, 2015). The inclusion of a GPS module allows for position updates (Chao *et al.*, 2010) and for important functionality such as waypoint implementation.

Distance sensors

Prominent distance sensors integrated with drones are light detection and ranging devices (commonly referred to as LiDAR and generally used for remote sensing tasks), sonar (used primarily to support navigation by detecting obstacles in the drone's flight path) and synthetic aperture radar (used for gathering texture data in remote sensing applications) (Pajares, 2015). Cameras as distance sensors also show promise, according to Floreano & Wood (2015).

[1]Image reproduced from – `https://developer.dji.com/mobile-sdk/documentation/introduction/flightController_concepts.html`

2.2 Digital image processing

Image processing forms the basis of computer vision systems, and according to Clouard *et al.* (2010), it can be organised under six categories, namely "image restoration, image enhancement, image compression, image reconstruction, image segmentation, and object detection". For the task of detecting invasive alien plants in drone sourced imagery, image processing algorithms for enhancement, segmentation and object detection are relevant. In this section, concepts and algorithms related to these tasks are introduced.

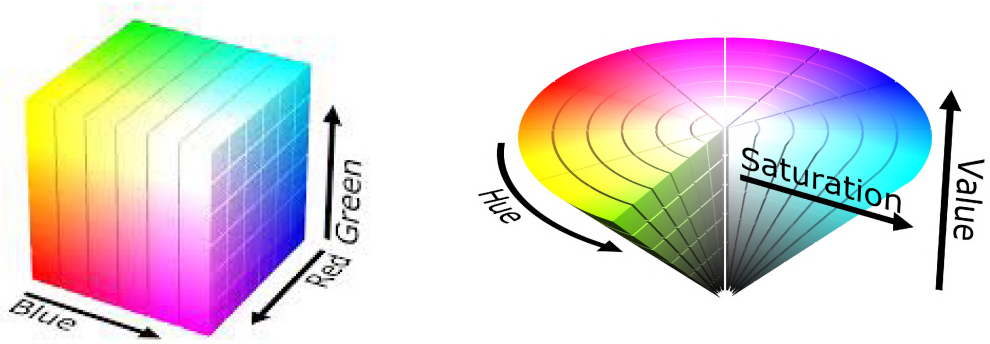
2.2.1 Image enhancement

The purpose of image enhancement "is to process an image so that (the) result is more suitable than (the) original image for (a) particular application", writes Kumar & Jaspreet (2017). Established algorithms for doing this are introduced.

Colour spaces

Colour spaces define a mathematical representation of colour, which includes chroma and brightness information. Within different colour spaces, certain features are easier to adjust than in others. Common colour spaces are briefly introduced below.

- Red Green Blue (RGB) – the RGB colour space can be arranged as a cube, with each axis dedicated to a colour, as seen in Figure 2.2(a). It is considered an "additive" colour space, where combinations of the three colours produce secondary colours.
- YUV – the intensity of the image, the Y channel, is separated from colour information, which is represented by the U and V channels. This colour space is often used for colour image processing when human perception is important.
- Hue Saturation Value (HSV) – in this colour space, the colour information is held separately to both the saturation and brightness (value) information. This model is more intuitive than the RGB space as colour information can be preserved while brightness or saturation is adjusted, or vice versa. The HSV colour space may be represented by a cone, as seen in Figure 2.2(b).

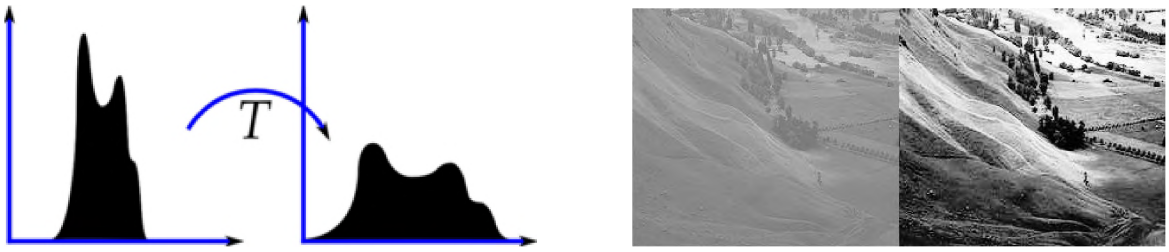


(a) RGB (b) HSV

Figure 2.2: RGB[2]and HSV[3]colour spaces.

Histogram equalisation

In each channel of an image (green, red, blue or grey) the pixel values can be visualised as a histogram showing their distribution and frequency. Greyscale images of low contrast tend to have narrow histograms and the process of histogram equalisation stretches these narrow histograms to cover the spread of all values, improving their contrast. This is shown visually in Figure 2.3.



(a) Equalisation process (b) Before and after equalisation

Figure 2.3: Histogram equalisation[4].

[2]Image reproduced from WikiMedia Commons – `https://commons.wikimedia.org/wiki/File:RGB_color_solid_cube.png`

[3]Image reproduced from WikiMedia Commons – `https://commons.wikimedia.org /wiki/File:HSV_color_solid_cone.png`

[4]Images reproduced from OpenCV Docs – `https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html`

2.2.2 Image segmentation

Image segmentation is the process classifying each pixel in the image to separate it into regions with similar characteristics and thus isolate regions of interest (ROIs) (Sonka *et al.*, 2014). This process allows for the transition in image analysis from pixel-based image analysis (PBIA) to object-based image analysis (OBIA), which is introduced in greater detail in Section 2.3.2.

According to Khan (2014), image segmentation techniques fall within the categories of edge-, threshold-, histogram-, region- and learning-based segmentation. It is noted that different algorithms are needed for segmentation of colour and greyscale images and that both the properties of an individual pixel or of a pixel and its neighbours may be considered (Khan, 2014).

- Edge-based segmentation – each pixel is classified as an edge or non-edge pixel, with those neighbouring pixels not separated by an edge given the same classification. In the survey conducted by Khan (2014), several edge-based image segmentation algorithms are introduced. The different approaches are a morphological watershed algorithm, edge detection for the approximation of the number of clusters to be used in K-means image segmentation, edge-based auto thresholding for multi-scale images and the use of a variance filter.

- Threshold-based segmentation – pixels are classified based on whether the pixel value falls within a specified range or not. That is, for a pixel with value p and threshold t, the classification is:

$$\begin{cases} 0 & p \leq t \\ 1 & p > t \end{cases} \tag{2.1}$$

  - Global thresholding makes use of a manually selected value as a threshold. All pixels values that fall on or below it are assigned one value, while all pixels that fall above it are assigned another.

  - Adaptive thresholding is used when different regions of the image have different brightnesses. The adaptive thresholding algorithm calculates thresholds for small regions of the image individually.

- Histogram-based segmentation – makes use of the frequency of grey levels in the image to separate the background and foreground. A widely used histogram-based

segmentation algorithm is Otsu's binarisation. This algorithm is useful when an image's histogram has two peaks, automatically selecting a value between them as a threshold value as in the third image in Figure 2.4.

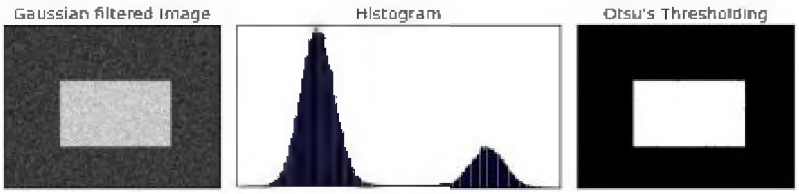


Figure 2.4: An illustration of the process of Otsu binarisation[5].

- Region-based segmentation – Assigning a classification to groups of neighbouring pixels with similar properties is termed region-based segmentation. Approaches covered in Khan (2014) used a combination of edge, region and spectral information in a morphological watershed algorithm and a least squares-based approach for faster separation and region growing.
- Learning-based segmentation – makes use of machine learning to separate the background and foreground, or object of interest.

2.2.3 Object classification and detection

Object classification is a process in which an object is assigned to one of a set of classes by a classifier based on its properties, which are represented as n-dimensional feature vectors (Sonka *et al.*, 2014). Object classification requires either matching algorithms or learnt pattern recognition algorithms to enable this classification.

In matching algorithms the grey pixel intensity values or features are used in a search for the location within an image of a known pattern. A match criterion is defined and each region and rotation in the image is assessed using this criterion. Subsequently, the local maximum of the evaluated criterion is found to give the location of the pattern in the image (Sonka *et al.*, 2014).

Learning algorithms, or statistical pattern recognition algorithms, take in a feature vector and output a class label for each object (Sonka *et al.*, 2014). These algorithms are split into supervised classification, which makes use of a representative subset of the data as a

[5]Figure reproduced from OpenCV Docs – `https://docs.opencv.org/3.4.3/d7/d4d/tutorial_py_thresholding.html`

training set to tune the parameters of the classifier in its learning stage, and unsupervised learning, such as cluster analysis, which has no training phase (Sonka *et al.*, 2014).

Object detection requires both the localisation of an object within the image, as well as its classification. Localisation can be performed using proposed candidate windows or through the use of object detection networks. Popular candidate proposal methods, which are suitable for all machine learning classifiers, are the sliding window approach, in which a window of fixed size proposes regions across the entire image, and object proposals, which attempt to reduce the search space. Object detection networks, such as You Only Look Once (YOLO) (Redmon *et al.*, 2016) or R-CNN (Girshick *et al.*, 2014), learn to locate objects by learning bounding box coordinates in addition to the class.

2.2.4 Feature extraction

For object classification or detection, it is necessary to extract features from the image or object of interest within it. Region description produces a numeric feature vector, which is affected by the resolution of the image (Sonka *et al.*, 2014). There are several different shape descriptors, namely contour-, region- and shape class-based. Region-based shape descriptors use geometric properties, while shape classes represent generic shapes.

The Hough Transform can be used for the detection of lines and other parametric shapes (Hough, 1962), making use of the representation of a line as a point when converted to an alternate parameter space, as explained in Sonka *et al.* (2014). By representing all edges in this space, the point with the most curves passing through it represents a line that passes through the most edges in the original xy space. This property can be used to detect lines or edges in images.

Histogram of orientated gradients (HOG) is another such method, which extracts features from an ROI within an image for object classification, the mechanics of which are detailed in (Xiao *et al.*, 2010). A gradient operator is applied to a region of interest and the resulting gradient vectors, consisting of both magnitude and direction, are binned to form a histogram. If this is done for several objects of the same type, a general histogram for an object can be developed.

2.3 Remote sensing and photogrammetric mapping

Remote sensing is the practice of analysing imagery captured at a distance by satellite or aircraft for the measurement of properties and characteristics on the earth's surface (Schowengerdt, 2007). A variety of different systems can be used, such as optical systems, hyper-spectral imagining and LiDAR systems. Drones have provided access to aerial images of unprecedented temporal and spatial resolution (Whitehead & Hugenholtz, 2014). These images can be input into the standard photogrammetric pipeline to produce outputs such as digital surface models (DSMs), digital elevation models (DEMs), structure from motion models (SfMs) and ultimately orthomosaics which can be further analysed through feature extraction (Nex & Remondino, 2014).

An orthorectified image is an image from which perspective distortions have been removed such that each point in an image appears as though the camera was right above it. An orthomosaic is a collection of orthorectified images which have been mosaicked to form one larger image. DEMs provide a mapped representation of the elevation above sea-level of the bare ground level within an area (Ajayi *et al.*, 2017), while DSMs provide a mapped representation of the elevation above sea-level of raised objects within an area (Ajayi *et al.*, 2017). DEMs are sometimes referred to as digital terrain models (DTMs), but this term can also refer to a representation of the DEM made up of points and contours. SfM software uses two-dimensional imagery to produce three-dimensional models and is robust even when there are large variations in scale (Whitehead & Hugenholtz, 2014). According to the authors, SfM models are likely to become the model of choice for drone conducted aerial surveys.

In remote sensing tasks, imagery captured at different wavelengths is often discussed, namely visual band, multi-spectral and hyper-spectral imagery. Visual band imagery contains only wavelengths within the visual part of the electromagnetic spectrum, while multi- and hyper-spectral imagery contains wavelengths in both the visual and infrared parts of the spectrum. Hyper-spectral bands are narrow over a continuous spectral range, unlike multi-spectral bands which are discrete, yielding greater spectral resolution (Colomina & Molina, 2014).

There are numerous examples of photogrammetric mapping and 3D modelling in the literature, such as applications for forestry and agriculture, mapping of sites and 3D reconstruction of structures for studies of archaeological sites, environmental surveying and traffic monitoring (Remondino *et al.*, 2011).

2.3.1 Photogrammetic software

In their review, Nex & Remondino (2014) describe a typical drone data acquisition and processing pipeline, which takes the form of mission planning, image acquisition, image triangulation, DSM/DTM generation and ultimately orthophoto and orthomosaic production. To assist in this process, there are several open source and many commercial photogrametric mapping software options available. Only those which are open source or commercial options geared towards DJI drones are introduced here.

OpenDroneMap[6] is an open source toolkit for processing aerial data obtained by a drone to produce a useful output. It can produce point clouds, DSMs, textured DSMs, orthorectified imagery and DEMs and is command-line driven.

AirPhotoSE[7] can produce rectified orthophotos, mosaics, DTMs and three-dimensional models. It does, however, require a map or orthophoto in addition to an image to provide control points.

VSfm[8] produces three-dimensional reconstructions, makes use of parallelism and is available via GUI or command-line.

DroneDeploy[9] is a commercial software aimed at DJI drones but is included because it offers a free 5cm/pixel resolution 30-day trial. It offers orthomosaicked maps, three-dimensional models, NDVI and volumetric analysis.

Burdziakowski (2017) performed a case study evaluation of OpenDroneMap compared to commercial software in which each was used to produce an orthophoto map. OpenDroneMap was tested on data obtained by an open source hardware helicopter while Pix4D Mapper was tested on data collected by the DJI Mavic Pro. It was found that OpenDroneMap can be used to produce geodetic grade models provided that a low distortion camera is used.

2.3.2 Pixel- and object-based image analysis

In remote sensing problems, it is frequently necessary to detect or classify objects within an image. This is often done using a learning algorithm, using features extracted from

[6] https://github.com/OpenDroneMap/OpenDroneMap
[7] http://www.uni-koeln.de/ al001/airphotose.html
[8] http://ccwu.me/vsfm/
[9] https://www.dronedeploy.com/

the image, which can be extracted at either pixel- or object-level.

In the PBIA paradigm features are extracted from each pixel, and a pixel-wise classification is made. The classification of the pixel depends only on the features of that pixel and not any of its neighbours. PBIA is best for mid-resolution imagery, where objects are typically the same size or smaller than a pixel (Calleja *et al.*, 2019). However, with the increased spatial resolution available, in which objects cover several pixels, OBIA has grown in popularity (Blaschke, 2010). OBIA is a two-step process, first requiring segmentation and then classification of each segment. Developed OBIA algorithms are not transferable between images, instead requiring the process to be applied from scratch on a new image.

2.3.3 Partitioning techniques

A crucial part of remote sensing is partitioning the image into objects, from which to extract features of interest. Common ways of doing this with reference to vegetation mapping and environmental monitoring are introduced below.

Vegetation indices

A well-established method in the analysis of remotely sensed images and orthomosaics is the use of vegetation indices. These are values calculated to exploit the different spectral properties of the image at different wavelengths, typically using the visual and NIR wavelength bands, to enhance certain features.

There are many categories of vegetation indices, which are listed by Harris Geospatial Solutions (2017) as broadband greenness, narrowband greenness, canopy nitrogen, canopy water content, dry or senescent carbon, leaf pigments and light use efficiency. The broadband greenness category is the most straightforward and yields a measure of the density of green vegetation. Within this category, there are a large number of different indices that can be calculated. The calculation of those considered relevant to our research as defined as follows, where R, G and B respectively denote the red, green and blue components of each pixel's colour value.

- Difference vegetation index (DVI)(Tucker, 1979):

$$DVI = NIR - R \tag{2.2}$$

- Normalized difference vegetation index (NDVI)(Rouse Jr *et al.*, 1974):

$$NDVI = \frac{NIR - VIS}{NIR + VIS} \tag{2.3}$$

- Enhanced vegetation index (EVI)(Huete *et al.*, 2002), with coefficients $G = 2.5$, $C_1 = 6$, $C_2 = 7.5$ and $L = 1$:

$$EVI = G * \frac{NIR - R}{NIR * C_1 * R - C_2 * B + 1} \tag{2.4}$$

There are also a number of vegetation indices that can be calculated using only the visible part of the spectrum.

- Excess green (ExG)(Woebbecke *et al.*, 1995):

$$ExG = 2G - R - B \tag{2.5}$$

- Colour index of vegetation extraction (CIVE) (Kataoka *et al.*, 2003):

$$CIVE = 0.441R - 0.881G + 0.385B + 18.78745 \tag{2.6}$$

- Excess green minus excess red (ExGR) (Camargo, 2004):

$$ExGR = ExG - ExR = ExG - 1.4R - G \tag{2.7}$$

- Normalised green-red difference (NGRDI) (Gitelson *et al.*, 2002):

$$NGRDI = \frac{G - R}{G + R} \tag{2.8}$$

- Vegetative (VEG) (Hague *et al.*, 2006), with reference value a=0.667:

$$VEG = \frac{G}{R^a B^{(1-a)}} \tag{2.9}$$

- Woebbecke index (WI) (Woebbecke *et al.*, 1995):

$$WI = \frac{G - B}{R - G} \tag{2.10}$$

- Combination (COM) (Guijarro *et al.*, 2011):

$$COM = 0.25ExG + 0.3ExGR + 0.33CIVE + 0.12VEG \tag{2.11}$$

- Combination 2 (COM2) (Guerrero *et al.*, 2012):

$$COM2 = 0.36ExG + 0.47CIVE + 0.17VEG \tag{2.12}$$

- Visible atmospheric resistant index (VARI) (Gitelson *et al.*, 2002):

$$VARI = \frac{G - R}{G + R - B} \tag{2.13}$$

- Green leaf index (GLI) (Louhaichi *et al.*, 2001):

$$GLI = \frac{(G - R) + (G - B)}{2G + R + B} \tag{2.14}$$

- Green chromatic coordinate (GCC) (Gillespie *et al.*, 1987)

$$GCC = \frac{G}{R + G + B} \tag{2.15}$$

Relative positioning

In cases where there are distinct relationships between the locations of objects, relative positioning can be used. Peña *et al.* (2013) used the positions of crop rows found from spectral characteristics, along with OBIA, to identify the location of weeds in a field, assuming that the weeds would not be within a row. A different approach was taken by Pérez-Ortiz *et al.* (2015), who used the relative positioning combined with a Hough transform to detect weeds between crop rows. Monteiro *et al.* (2019) note that the Hough transform is often used to detect crop rows. The use of relative positioning combined with vegetation indices was also used with reference to surveying vineyard health, in which two cultivars were identified by different row spacing (Remondino *et al.*, 2011).

2.3.4 Photogrammetric processing challenges

Photogrammetry using drone imagery has a number of challenges to overcome, as discussed by Whitehead & Hugenholtz (2014). The high-spatial resolution obtained by low altitude flying requires many more images to map the same area than would be required by lower resolution satellite imagery. Whitehead & Hugenholtz (2014) discussed how the high number of images can lead to artefacts or distortions when creating an image mosaic and that the very-high-resolution imagery makes it unsuitable for the PBIA workflow, as individual plant features are visible instead of being contained within a single pixel. In addition, the high number of images used also require automation for the process of orthoimage generation (Remondino *et al.*, 2011).

Furthermore, it is possible for variation in brightness across a mosaic to occur, which leads to difficulties when only spectral features are used in analysis (Whitehead & Hugenholtz, 2014). Shortcomings, such as the lack of a NIR band and image vignetting, further complicate spectral analysis (Whitehead & Hugenholtz, 2014). The lack of a NIR band makes detecting vegetation more difficult as vegetation indices cannot be calculated, while image vignetting results in the edges of an image being darker than the centre of the image, giving a false variation in brightness.

2.4 Machine learning platforms

Learning algorithms determine general patterns in data in an automated fashion from provided examples (Sonka *et al.*, 2014). Statistical classifiers take in the provided data, which is comprised of features, and output a single value denoting the predicted class (Sonka *et al.*, 2014). The data provided as examples to the learning algorithm is comprised of sets of features which describe it, called "training vectors". During the training phase, in which the training vectors are supplied to the classifier, two approaches can be used: supervised learning or unsupervised learning. In the case of supervised learning, each of these training vectors is accompanied by a known class label whereas in unsupervised learning this is not supplied, instead using clustering in the data to form classes. Some examples of unsupervised learning methods are K-means, principal component analysis and self-organising maps, whereas examples of supervised learning methods include k-nearest neighbours (KNNs), support vector machines (SVMs), decision trees, random forest classifiers and neural networks (Kung, 2014). These supervised methods are introduced in greater detail in the next section.

2.4.1 Supervised machine learning algorithms

K-nearest neighbours

One of the simpler machine learning algorithms, the KNN algorithm uses n-dimensions, each of which represents a feature (Kotsiantis *et al.*, 2007). Each of the classes is represented by a set of these features, which are established through training. To classify an input, the Euclidean distance between the input point and k nearest points of the other classes is determined, with the class being that of the majority of closest points. When $k = 1$ the algorithm is simply referred to as the Nearest Neighbour algorithm.

Support vector machine

The main objective of SVMs is to find the decision boundary that separates classes by the largest possible margin. Of all training vectors, SVMs use only those which are entirely necessary to determine class separation. These critical training vectors are called support vectors (Kung, 2014).

Initially, SVMs were for binary classification only, with the output being either positive or negative (Yu & Kim, 2012), but they were expanded to multi-class through the coupling of multiple binary class SVMs. There are now many variants in the SVM classification family; the main groups allow classification or regression and linear or non-linear separation of classes. Further important subgroups include latent, ranking, exemplar and one class SVMs.

Linear SVMs separate classes with a hyperplane (Yu & Kim, 2012), while non-linear SVMs are more complex, using a non-linear kernel function to determine the decision boundary (Kung, 2014).

A one-class SVM is typically used when there are few or no negative samples available, which makes determining the decision boundary more difficult than for multi-class SVMs. The smallest hyperplane surrounding the positive samples must be found, such that it maximises the number of positive samples that fall into it while minimising the chance for negative samples to be accepted (Khan & Madden, 2009).

In an exemplar SVM, each positive element (exemplar) has its own linear SVM, which distinguishes the positive element from all negative training elements. These exemplar SVMs are grouped together to form an ensemble, and an object is determined to be a

member of the positive class if it is classified to belong to any of the exemplar SVMs. This SVM variant is "defined by a single positive instance and millions of negatives", according to Malisiewicz *et al.* (2012).

A latent variable is one which is not directly observed but instead inferred from other observed variables. Latent SVMs are multiple instance SVMs that make use of latent variables (Felzenszwalb *et al.*, 2010b).

Decision trees

A decision tree is composed of a collection of nodes, each of which performs a feature test on its input. This process divides the output into subsets until a leaf node is reached and a classification is assigned (Zhou, 2012).

Random forests

Random forests are an ensemble of decision trees for classification of a target, the final classification of which is decided by majority vote of the individual classifiers (Osman, 2010). A random subset of features is used to construct each node of the decision trees (Zhou, 2012). Random forests minimise the generalisation error instead of the training error, unlike most machine learning algorithms (Osman, 2010).

Cascade classifiers

Cascade classifiers are composed of several stages of simple classifiers through which an ROI is passed. At each stage, the ROI is either rejected or accepted (passed) and is finally classified as part of the target class if all stages of the classifier are passed (OpenCV, 2017). Cascade classifiers decrease the complexity of the classifier but require a 'heavier' training algorithm (Vedaldi *et al.*, 2009).

Deformable part-based model

A deformable part-based model makes the assumption that an object is constructed from a set of parts, which are in a deformable arrangement and hence their locations are unknown, so position is a latent variable (Felzenszwalb *et al.*, 2010b). A significant part, the 'root', is first searched for and upon determining possible locations of this root part,

the positions of the remaining parts are found for each possible root part (Felzenszwalb *et al.*, 2010b). Using a sliding window the HOG features of the window are extracted and thereafter a filter is applied to these features (OpenCV, 2015).

This model can be sped up through the use of a cascade classifier, which removes low scoring (partial) hypotheses through thresholding and focusses on high scoring hypotheses, according to (Felzenszwalb *et al.*, 2010a).

Neural networks

A system of connected processors, called neurons, forms the basis of the well known neural network algorithm. Each of the connectors has a weight value and the process of supervised learning by gradient descent adjusts these values in the training stage to make the network develop a particular classification behaviour towards a dataset (Schmidhuber, 2015). In the network, a neuron is activated from either an external input to the system or through the weighted values of incident connectors. The output of a classification network is usually a single class label.

A variant of the neural network is the convolutional neural network (CNN). The modern-day CNN emerged from the work of LeCun *et al.* (1990) for handwritten digit classification from images, making use of supervised learning and back-propagation. However, it was not until the now-famous AlexNet architecture by Krizhevsky *et al.* (2012) that the modern-day CNN was popularised. AlexNet was trained using images from the ImageNet Large Scale Visual Recognition Challenge (Russakovsky *et al.*, 2015) database and gave one of the first indications that deep learning CNNs would overtake shallow learning methods as the state-of-the-art on image classification challenges. The AlexNet architecture consisted of eight learned layers; five convolutional layers with max-pooling layers, three fully connected layers and ending with a 1000 class softmax function, along with dropout regularisation. The network had 60 million parameters and 650 000 neurons and achieved a top-1 accuracy of 57.2% and top-5 accuracy of 80.3% on the ImageNet dataset, as reported by Iandola *et al.* (2016). Subsequent to this, deep learning-based architectures have held onto the title of state-of-the-art, with architectures such as ZF Net, VGG Net and GoogLeNet notable architectures with strong performance over time on the classification challenge (Russakovsky *et al.*, 2015).

To achieve better accuracy on classification challenges, the AlexNet architecture was modified to produce the ZF Net and VGG Net. According to Zeiler & Fergus (2014), ZF Net used 7×7 filters as well as a deconvolution network as a diagnostic tool to improve accu-

racy, while Simonyan & Zisserman (2014) produced the VGG Net which used 3×3 filters. Later models began to make use of 'skip connectors', which allowed the passage of information deeper into the network. The use of 'modules' allowed for a specific organisation of layers to be repeated easily within the architecture. GoogLeNet, created by Szegedy *et al.* (2015), made use of 'inception modules'. These modules had skip connectors that were repeated nine times within the network along with 1×1 filters. The Microsoft Residual Network (ResNet) made further use of skip connectors to allow deviations from identity layers, which are layers that would otherwise not affect results due to accuracy saturation and simply transfer their input to their output (He *et al.*, 2016).

The above architectures aimed to achieve the best possible accuracy but resulted in models that were ever larger and hence difficult to deploy on memory impoverished devices (Iandola *et al.*, 2016). Another school of architectures was designed to minimise their size without significantly decreasing their accuracy, notably the SqueezeNet and MobileNet.

Squeezenet, developed by Iandola *et al.* (2016), is notable for its success in this, having achieved the same level of accuracy as Alexnet, but with 50× fewer parameters and 510× smaller model size. The architecture made use of 1×1 filters instead of 3×3 to reduce computation and the 1×1 squeeze layer reduced the depth of the network, before increasing it again. Downsampling was done at a late stage so as to maintain large activation maps in convolutional layers. The core of the SqueezeNet architecture is the use of 'fire' modules, each of which comprises a 1×1 convolutional squeeze layer, followed by a mix of 1×1 and 3×3 filters in an 'expand' layer.

Another model notable for small size and good performance is the MobileNet family of lightweight architectures, which are optimised for latency and intended for use in mobile and embedded applications, as reported by Howard *et al.* (2017). The architecture makes use of depth-wise convolutions, which reduce computation in early layers but with only a small impact on accuracy. A full version of MobileNet was able to achieve accuracy between that of GoogLeNet and VGG16 at 70.5%. It was also 32× smaller than VGG and nearly 2× smaller than GoogLeNet, while also being significantly less compute-intensive than either model: 27× and 2.5×, respectively. A reduced MobileNet, with model size comparable to SqueezeNet, achieved 4% better accuracy than Squeezenet and was 22× less computationally expensive (Howard *et al.*, 2017). Having accuracy competitive with that of full-sized models, but with the size of a reduced model makes the MobileNet family of architectures strongly competitive.

A modification from the standard classification network, replacing dense layers with con-

volutional ones, allows for semantic segmentation – a pixel-wise classification of the entire image (Long *et al.*, 2015). These networks are called fully convolutional networks (FCNs). Existing architectures can be adapted to the FCN architecture and combined with a skip architecture which combines deep, coarse information with shallow, fine information to produce segmentations. The replacement of fully connected layers with convolutional layers allows networks to output heatmaps, which can be thresholded to produce binary segmentations.

Transfer learning and fine-tuning It is possible that the dataset available for training a CNN may be too small to train it to convergence. According to Nogueira *et al.* (2016), an alternative approach to training, in this case, is a process called fine-tuning, or transfer learning. Networks trained in this manner make use of the weights of an existing architecture that has already been trained to convergence on another dataset as a starting point, before training further using a new dataset. This is made possible as the early layers of a CNN tend to learn general low-level features that are largely dataset independent, such as edge and blob detection. Nogueira *et al.* (2016) also state that transfer learning has the additional benefit of providing better performance even than that attainable through augmentation of a small dataset, as well as being significantly faster than training a network from scratch.

It is possible to utilise transfer learning to various degrees. In some cases, the pre-trained weights are used for initialisation only and thereafter all weights are adjusted, whereas in other cases, low-level layers are 'frozen' with their learning rate set to zero and only the final layers are adjusted. In the latter situation, a low learning rate is used.

Feature maps

- Activation maps

  At each layer in the network, it is possible to visualise the activations in each filter for a particular input image as an activation feature map. Each filter learns different properties of the image and generally early layers detect general features, while later layers learn to detect finer detail. By visualising the activation feature maps it is possible to compare them with the original image to find out which features the network learns for that particular filter. It is most common to visualise features in early layers where it is possible to project back to the pixel space and to use the image as a diagnostic tool to adjust the network architecture to be more effective,

as notably done by Zeiler & Fergus (2014).

- Class activation maps

  Class activation maps highlight the region of an image which allows the CNN to make a distinction between the classes. As explained in Zhou *et al.* (2016), the class activation map is produced by summing the filters in the global average pooling layer and weighting each by its weight in the final softmax layer. This produces a 'heatmap' showing the region of the CNN's attention, which can be used for localisation of the discriminative part of the image while only having been trained with class level labels.

2.4.2 Model evaluation

An important step for assessing how well a model will generalise to unseen data is through model evaluation. In this subsection, we introduce metrics to quantify model performance and the k-fold cross-validation technique for assessing the model on different portions of the dataset.

K-fold cross-validation

Cross-validation is a method of resampling without replacement to determine whether a model generalises well on unseen data and whether its performance is independent of the samples in the validation set. As explained by Raschka & Mirjalili (2017), in k-fold validation the pool of data available for training is divided into k segments or folds. Each of these folds is, in turn, held out of the training process and used as the validation dataset. For each fold, an estimate of performance is produced, which gives insight into whether the model performs well on the unseen data or whether the performance is dependant on the composition of the training and validation sets. Once cross-validation has been completed and metrics obtained, the model can be retrained on the entire training set to produce a model for inference. According to Kohavi (1995), it is standard to do 10-fold validation, although small datasets can have more folds. They go on to say that stratified k-fold cross-validation gives a slight improvement on standard k-fold cross-validation, as it maintains the proportions of each class for each of the folds.

Metrics

A number of metrics are important for evaluating model performance. Those used are introduced in this section. The predictions of a network can be analysed in a 'confusion matrix', in which the true positives (TP), false positives (FP), false negatives (FP) and true negatives (TN) are displayed. True positives are those instances of the target class that were classified as such, while true negatives are those instances that were correctly classified as not of the target class. False positives are then instances not of the target class incorrectly classified as of it, while false negatives are the inverse of this.

Accuracy – the total number of pixels that were predicted correctly, as a fraction of the total number of pixels in the image n, which is equal to $TP + TN + FP + FN$.

$$Accuracy = \frac{TP + TN}{n} \tag{2.16}$$

Precision – the fraction of positives detected correctly of all predicted positives.

$$P = \frac{TP}{TP + FP} \tag{2.17}$$

Recall – the fraction of positives detected correctly of all true positives.

$$R = \frac{TP}{TP + FN} \tag{2.18}$$

F1 score – provides a balance between precision and recall, particularly when there is a class imbalance.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall} \tag{2.19}$$

2.4.3 Annotation

Supervised learning requires a set of inputs with corresponding ideal outputs. The composition of these annotations depends on the task. In machine vision classification problems, a single label to denote class is needed for each image in the training set, while object detection requires position information as well as class. It is common to place a bounding box around the object of interest within the image, recording its class information as well.

Annotations can be saved in different formats, such as PASCAL Visual Object Classification and YOLO format. These annotations can be used to extract objects as sub-images for classifier training from the larger image. Semantic segmentation, on the other hand, requires a 2D annotation mask, indicating the class of each pixel within the image.

Various software exists to assist in annotating images for supervised learning. LabelImg[10] allows images to be saved in both PASCAL Visual Object Classification and YOLO format, providing annotations suitable for object detection. Vatic (Vondrick *et al.*, 2013) is a tool for video annotation, that assists annotators through interpolation of bounding boxes between key frames. Vatic also offers an option to crowd source annotations using Amazon's Mechanical Turk.

Scale, position and aspect ratio

A downside of using sub-images as training data is that the scale and aspect ratio of the objects can vary. Since a set of consistently sized vectors is required as training vectors for the machine learning algorithm, all images need to have the same dimensions. This can be achieved by cropping the images to the correct dimensions, cropping the images to a standard aspect ratio, zero-padding images to obtain the correct aspect ratio or including neighbouring pixels to obtain the correct aspect ratio and then resizing the images to a standard size.

When assessing a classifier on an image for object detection, it is necessary to search the entire image for the object. This is often done using a sliding window to which a classifier is applied when searching for an object. It is also common practice to apply a sliding window to different scales of an image, as objects may be closer or further in the view field.

Andelson *et al.* (1984) discussed pyramid scaling as a method for efficiently detecting features in multiscale images. A pattern is kept at a fixed scale while the original image size is decreased by a factor of two several times to form a 'pyramid' of image scales. The pattern is then convolved with the image at all scales of the image pyramid.

[10]https://github.com/tzutalin/labelImg

2.5 Summary

This chapter introduced concepts important to this research. The term drone was defined to refer to small unmanned quadcopters, and important topics related to unmanned aerial systems were introduced. Relevant digital image processing techniques were highlighted, and a brief introduction to remote sensing and photogrammetric mapping was delivered. Finally, appropriate machine learning algorithms and concepts were introduced to provide a fuller background to the methods used in this research.

In the following chapter, an overview is provided on the methods and strategies conducted in related research. This chapter also provides insight into how our research fits into the greater field of drone-based conservation efforts, and with regard to the detection of vegetation.

Chapter 3

Related Work

This chapter discusses related research, particularly that of related drone applications in conservation and research, as well as techniques for detection of objects in aerial imagery. The related research in the latter section is grouped according to the target for detection, namely wildlife, precision agriculture and remote sensing applications. Through the discussion of this related work, insight is provided into the niche in which the research in this thesis fits.

3.1 Drone applications in related research

Drones have become a valuable research tool, providing an efficient way of obtaining data (Marris, 2013), particularly in areas that are inaccessesible (Chabot & Bird, 2015)(Nex & Remondino, 2014), dangerous to access or approach (Chabot & Bird, 2015)(Nex & Remondino, 2014), sensitive to disturbance (Chabot & Bird, 2015) or areas that are too large for a ground based team to cover (Patton, 2013).

Within the literature, there are many examples of the uses of drones for particular research tasks. Broadly these may be grouped to focus on wildlife and habitat surveying and monitoring for both research (Chabot & Bird, 2015) and conservation (Jiménez López & Mulero-Pázmány, 2019), mapping applications, which according to Nex & Remondino (2014) can themselves be grouped into applications in "agriculture, forestry, archeology and architecture, environment,emergency management and traffic monitoring", as well as inspection, surveillance and 3D modelling (Remondino *et al.*, 2011). Drones have also been used to monitor landscapes of interest, such as rangelands (Rango *et al.*, 2009) and

riparian thickets (Habel *et al.*, 2018).

Nowak *et al.* (2019) reviewed drone uses in environmental biology and observed, based on the papers reviewed, that drone use is likely to continue to increase over the next few years. Most studies were targeted at measuring vegetation parameters, such as "crown height, volume, (and) number of individuals", "quantification of spatio-temporal dynamics of vegetation changes" or for census studies of animals or birds.

Drones have proved to be particularly valuable in remote sensing and photogrammetry related research, providing access to temporal and spatial resolutions that are otherwise unattainable (Whitehead & Hugenholtz, 2014) due to the high cost of manned flights (Remondino *et al.*, 2011) (Nex & Remondino, 2014) and insufficient satellite resolution for some tasks (Whitehead & Hugenholtz, 2014). Drone imagery has been used to produce GIS products like DTMs and DSMs for 3D mapping applications (Nex & Remondino, 2014) as well as DEMs (Ajayi *et al.*, 2017). There has been much research in the remote sensing community in the past few years, of which the mapping and monitoring of vegetation forms only a small part. For a comprehensive overview of the use of drones in remote sensing applications, please see (Pajares, 2015).

Besides advantages in resolution, Cruzan *et al.* (2016) note that small drones can collect large volumes of data with little effort and drone surveys cause less disruption to sensitive habitats than ground-based surveys. Small drones with cameras capable of facing 90 degrees from the horizontal are suitable for distribution and abundance surveys of individual species and for producing accurate vegetation maps over large areas. A challenge faced, however, is that trees and shrubs can block other vegetation from view.

The availability of increased spatial resolution from drone sourced imagery is useful for conducting censuses of endangered plants within fragile habitats (Rominger & Meyer, 2019). This, along with the increased temporal resolution available allows for "responsive, timely and cost-effective monitoring of ecological processes", which was shown by Ventura *et al.* (2018) who mapped three ecologically sensitive coastal environments using RGB imaging and an OBIA workflow. Furthermore, in his study of small-scale habitat fragmentation, Yi (2016) demonstrated that drones allow for a stable, robust and efficient way to assist in monitoring and analysis.

For wildlife studies, Chabot & Bird (2015) observe in their review that "wild animals tend to be elusive, wide-ranging, sensitive to human disturbance, and (or) dangerous to approach". Drone surveys help to address these issues, providing a valuable data gathering tool for both researchers and conservationists (Marris, 2013, Patton, 2013).

The detection of wildlife is necessary both for research purposes and to obtain data for use in conservation management and protection.

In a bid against poaching, drones have been used for the detection and tracking of both poachers and animals. An adaptive visual tracking algorithm was introduced by Olivares-Mendez *et al.* (2015) as an autonomous method of monitoring animals in the wild, while Koh & Wich (2012) noted that a conservation drone could be used to survey large animals. Wildlife is not always directly detectable from the air if its habitat is dense vegetation, but a study by Van Andel *et al.* (2015), in which drones were used to monitor chimpanzee nests, indicates an indirect method of monitoring their presence. Drones have also been used to identify the actual poachers, as opposed to their target animals, such as the inclusion of facial detection by Olivares-Mendez *et al.* (2015) as part of their autonomous anti-poaching system, to develop a poacher database.

Drone imagery has also been used for population counts, particularly for plant and animal censuses, such as those described in Cruzan *et al.* (2016) and detailed in Section 3.2.3. Recently even the movement of invasive insects has been monitored using drones and ultraviolet lighting technology (Stumph *et al.*, 2019).

3.2 Detection techniques in aerial imagery

A crucial aspect of many drone applications for both ecological research and conservation is the detection of targets in the collected imagery. In this section, an overview of relevant approaches in wildlife studies, precision agriculture, plant mapping and plant censuses is given.

3.2.1 Wildlife studies

In the review by Chabot & Bird (2015) various uses of drones for wildlife studies are reported. With reference to the detection and/or counting of animals, the works of Abd-Elrahman *et al.* (2005), Selby *et al.* (2011), Chabot & Bird (2012), Grenzdörffer (2013), Maire *et al.* (2013), Christiansen *et al.* (2014) and Van Gemert *et al.* (2015) were particularly mentioned. We report on the methods used by these authors for this purpose.

Abd-Elrahman *et al.* (2005) used a combination of feature- and area-based analysis to exploit colour and shape in a multi-stage pattern recognition algorithm for bird detection.

Normalised cross-correlation matching was performed between a template and a sliding window, and region grouping was then performed using grey pixels where there was a high cross-correlation coefficient. These grey pixels were used as seed pixels around which to grow regions. The maximum grey values in a 3×3 batch at the centre of the regions were then compared to the average of a 3×3 batch in the bird template, and a threshold used for the difference between these. The area was computed for the remaining candidate regions to remove small polygons created by noise. This method worked well for birds that were shifted in the x- or y-directions in relation to the template, but not as well for those that had undergone scale or rotational transformations. Chabot & Bird (2012) also investigated bird detection in drone imagery, comparing aerial image counts to ground counts and showed the potential for using drone imagery in this manner for surveys of geese.

Selby *et al.* (2011) developed an algorithm for the detection and tracking of marine animals, particularly whales. This system was designed for computationally impoverished systems, such as drones, and objects set against relatively homogeneous backgrounds. Images were segmented based on colour and pixel value heuristics hue and saturation. First, a target colour model was developed. This was done by converting a frame containing the object to the HSV colour format and computing a histogram of the hue and saturation values of the image. A user then manually set a threshold for the maximum and minimum hue and saturation values by selecting the area containing the object and subsequently, all values outside of these threshold values were set to zero in the histogram, resulting in a target colour model histogram. Object detection for subsequent frames in a video was then implemented by converting each frame to HSV and backprojecting the hue and saturation values through the histogram to produce a probability image. Erosion followed by dilation was then performed on the image to remove small false positives and pixel groups were classified into regions. The largest region was then identified and a bounding box plotted around it. This algorithm was found to be robust and acceptable for computationally impoverished systems, but it requires that the target fills most of the image and that the background is fairly homogeneous.

Grenzdörffer (2013) performed an evaluation of automatic counting as a census technique for gulls in a colony, using multi-spectral imaging and a mosaic. Studies were conducted over two years, using a very similar methodology, but with perfect conditions and a slightly higher flight altitude in the second year. This resulted in better image quality, and as a result, a better classification accuracy. The methodology used made use of the relatively unique colouring of the gulls to identify a set of potential bird objects, from which a minimum and maximum area threshold were applied to remove unlikely

candidates. Due to variation in the colouring of plumage across the birds, two objects were often detected for a single bird, so these regions were merged using a maximum distance criterion. Another minimum area criterion was applied to the resulting objects. Performance of the algorithm was assessed using a subset of the data and comparing it against a visual count.

The use of thermal imagery to detect wildlife in agricultural fields was investigated by Christiansen *et al.* (2014). The images were dynamically thresholded to detect hot objects, and a thermal heat signature was produced using morphological operations and parametrised using the discrete cosine transform. Thereafter, a KNN classifier was used to discriminate between animals and non-animals.

Van Gemert *et al.* (2015) investigated an automatic methodology for wildlife detection and counting, validated through cow detection. Three algorithms suited for running onboard the drone were investigated, namely two deformable part-based models and an exemplar SVM, with the last producing the best results.

An automated algorithm for dugong detection was developed by Maire *et al.* (2013), which made use of both colour and morphological features to determine ROIs. The image was thresholded for colour rarity, high image entropy and high red-ratio (red pixel value over the sum of blue and green pixels) and the intersection of these three images was used to determine ROIs. Blobs within the ROIs were filtered by size and an extended maxima transform was applied to remove all peaks that differed from those around them by less than a threshold value. The final stage involved Otsu thresholding each quadrant of the ROI, to account for variation in colour caused by variable depth of the dugong below the surface of the water, merging the results and creating a feature vector of the resulting blob. This was then used to determine how close to elliptical the blob was, as the predominant shape of a dugong is elliptical. This algorithm proved robust with respect to illumination, due to segmentation by rare colour and the extended maxima transform, but performed less well when high noise was present due to breaking waves. Maire *et al.* (2013) recommended that a learnt shape classifier would be more effective than the current shape filtering module.

More recently, the same task of dugong detection has been found to be more successful using region proposals and deep CNNs (Maire *et al.*, 2015). A simple linear iterative clustering algorithm was used to generate superpixels, which were then used to extract sub-images from the original image, and these were fed to the CNN. Data augmentation (rotations and scaling) was used as well as hard negative mining. The authors note that as the images were taken at a known height, no pyramidal analysis was required.

Other recent studies were conducted by Rey *et al.* (2017), Seymour *et al.* (2017), Hodgson *et al.* (2018) and Hong *et al.* (2019). A brief overview of each of these is given.

According to Rey *et al.* (2017) large mammals can be detected in African Savanna biomes using an ensemble of exemplar SVMs as a classifier on RGB drone imagery. Annotations were crowd sourced and an annotation confidence map produced, with ground truth annotations selected as those areas where at least half of the annotators tagged it. A histogram of colours and bag of words were used as features.

Seymour *et al.* (2017) took a different approach to the other studies discussed and mapped grey seal colonies using thermal imagery collected by drones. RGB and thermal orthomosaics were produced, with areas of similar ambient land temperature to that of the seals masked out. Seals were discriminated using "temperature, size and shape of thermal signatures". In the case where seals were packed into close groups, edge detection was used to separate individual members.

Hodgson *et al.* (2018) found that manually counting birds in drone sourced aerial imagery resulted in substantially better accuracy than counting through traditional ground-based techniques. A colony of fake birds of known number were used. Furthermore, they found that a linear SVM could be utilised to perform these counts semi-automatically, achieving 94% similarity with the manual counts in the aerial imagery. The SVM was trained using a PBIA approach, with each pixel represented using an invariant Fourier histogram of oriented gradient features. The SVM produced likelihood maps which were thresholded to determine bird locations.

A number of object detection networks were evaluated by Hong *et al.* (2019) on the task of detecting birds in drone imagery, namely: Faster Region-based Convolutional Neural Network (R-CNN), Region-based Fully Convolutional Network (R-FCN), Single Shot MultiBox Detector (SSD), Retinanet, and YOLO. Faster R-CNN was found to produce the best accuracy, while YOLO was the fastest.

When comparing earlier works, it is noted that the colour space used was an important factor for segmentation of the image, with grey pixel values, hue and saturation values in the HSV colour space, rare colours and red-ratios used in the respective works to determine areas of interest. Minimum area was also important and was used to filter out areas that were too small to be candidates by Abd-Elrahman *et al.* (2005), Grenzdörffer (2013) and Maire *et al.* (2013). The algorithm developed by Selby *et al.* (2011) had the benefit of being computationally light and as such suitable for use on a drone on-board processor. With the exception of Seymour *et al.* (2017), works from 2014 onwards all

use machine learning. The improvement found by Maire *et al.* (2015) compared to their previous work indicates that deep learning can be used to improve classification. The work conducted by Hong *et al.* (2019) suggests a different approach to the independent selection of ROIs and classification, instead unifying these two steps using networks for object detection.

3.2.2 Precision agriculture

A popular use for drones is the monitoring and mapping of weeds in crop fields for 'precision agriculture' (Hodgeson, 2013). Although this is not strictly a conservation effort, the methodology involved in detecting and mapping these weeds is the same as with detecting any other vegetation type.

The process of segmenting an image into plant and background classes is an important step prior to the classification of plants into weed or non-weed classes. According to Hamuda *et al.* (2016), the three main ways of performing this segmentation are the use of vegetation indices (referred to as colour indices in their paper), threshold-based methods and learning-based methods. Complex backgrounds proved to be a challenge, as were variations in illumination, particularly those between sunny and overcast days and the presence of shadows (Hamuda *et al.*, 2016). Vegetation indices were observed to be suitable for real-time applications, while threshold- and learning-based segmentation techniques were less so. However, unlike vegetation indices, threshold-based segmentations could be adjusted for different illumination conditions, and learning-based methods could be trained to operate under these conditions.

Vegetation indices are frequently employed as a baseline for comparison against more complex proposals (Hamuda *et al.*, 2016). The indices NDVI and ExG are particularly prominent for segmenting soil from vegetation (Monteiro *et al.*, 2019), with the former index requiring a NIR band in addition to the visual band required in each. Further classification is necessary to differentiate weeds from crops (Abouzahir *et al.*, 2017).

Pérez-Ortiz *et al.* (2015) found that a semi-supervised approach led to the best weed maps within a sunflower crop plot. Different classification approaches, unsupervised, semi-supervised and supervised were compared, as well as altitudes of 30 m, 60 m and 100 m at which the drone sourced imagery was collected. Both a multi-spectral and visual spectrum camera were used, and orthomosaics were created. In addition to the NDVI and ExG vegetation indices, the Hough transform was utilised to detect crop rows, which

was found to improve accuracy when used as an extra feature. Both altitudes of 30 m and 60 m were found to be acceptable, with 30 m slightly better than 60 m.

Peña *et al.* (2013) also made use of drone sourced multi-spectral imagery at 30 m and an OBIA approach to mapping weeds in an early-season maize field. Crop row orientation was estimated from segmentation outputs, and NDVI was then used to separate bare soil from vegetation. Thereafter, maize was distinguished from weeds by location below or above buffer zones along crop rows and a grid structure was produced, showing the level of weed infestation per grid block between crop rows.

Lopez-Granados (2011) observed that multi-spectral imagery (with visible and NIR bands) with pixel size of 0.5 m can provide accurate weed maps, but that larger pixel sizes negatively affect discrimination ability. The author contends that hyper-spectral resolution is necessary to discriminate weed species and notes that there is spectral variation between different phenological stages and that late detection may increase these compared with early detection.

Further investigation as to what spatial and spectral properties of imagery were needed for weed seedling discrimination in an infested sunflower field was conducted by Torres-Sánchez *et al.* (2013). A drone carrying both a visual camera and multi-spectral camera at altitudes of 30 m, 60 m and 100 m were used to evaluate this. Images were orthomosaicked and it was found that spectral differences between weeds, crops and soil were most readily apparent at 30 m and that the NDVI index provided the most discernible difference between vegetation and soil. In a later work, Torres-Sánchez *et al.* (2014) showed that spectral indices using only the visible part of the spectrum, particularly ExG and VEG, could be used for mapping the vegetation fraction in early season wheat fields using visual spectrum drone imagery.

A study comparing pixel-based, object-based and a mixed approach for detecting thistle weeds in a cereal crop was conducted by Franco *et al.* (2018). The imagery was collected by drone at 50 m and annotated both pixel- and object-wise using their RGB intensities, with objects having the value of the average RGB intensities of all pixels within them. An object was annotated as part of either the weed or cereal class if more than half the pixels within it fell into that category, which meant that not all pixels within an object belonged to that class. When considering the mixed classification approach, a soft accuracy measure was computed for object-level annotations from pixel-level classifications, using the proportion of weed pixels to all pixels within the object. This approach was found to be more successful than a pure pixel- or object-based analysis. The authors relied mainly on the use of a KNN classifier, but also briefly examined single hidden layer

feed-forward neural networks, which they found to give excellent performance, particularly in terms of sensitivity. Monteiro *et al.* (2019) reviewed papers for weed mapping in aerial imagery and found no tendency in the works reviewed to favour either PBIA or OBIA approaches.

The use of a CNN for weed-crop classification was demonstrated by Milioto *et al.* (2017), to distinguish sugar beet crops from weeds. This approach made use of an NDVI mask to segment vegetation from soil as well as a connected blob algorithm to identify individual plants. These individual plants were then extracted as sub-images and fed to the CNN, which was trained in an end-to-end fashion such that no hand-crafted features were used. Exceptional accuracy was achieved using this approach and the trained CNN was also found to be suitable as a starting point for fine-tuning to other applications and later growth stages of the crop.

In a comprehensive review of deep learning in agriculture, Kamilaris & Prenafeta-Boldu (2018) observed that deep learning methods almost always out-perform other image processing techniques. CNNs are particularly common in surveyed papers, with architectures such as AlexNet, VGG16 and Inception-ResNet used. The initialisation of the network with pre-trained model weights or the use of a fine-tuning approach was sometimes used to assist training with small datasets, as well as data augmentation, which was a commonly used approach. Pre-processing of images prior to classification was also common, with common tasks being the resizing of images, image segmentation, background removal and foreground extraction, changing to the HSV colour space or orthomosaicking. Common issues in weed detection were identified to be illumination, resolution, soil type and insufficient discernible variation in characteristics between weeds and crop.

3.2.3 Plant species census/mapping in remote sensing

Plant species mapping is important from a conservation perspective for monitoring populations that are invasive or endangered. With the unprecedented spatial resolution provided by drone sourced aerial imagery (Whitehead & Hugenholtz, 2014) a shift in image analysis techniques has been seen from PBIA to an OBIA approach, as described in (Blaschke, 2010). This means that in vegetation detection tasks, pixels in images no longer contain entire plants which are instead represented by multiple pixels.

This does not necessarily mean that PBIA for image analysis should be abandoned. In a study comparing pixel-, object and hybrid-based approaches to invasive plant species mapping, Dvorák *et al.* (2015) found that PBIA was still better than an OBIA approach

when plants formed dense stands in which individual plants were indistinguishable. However, they did find that an OBIA approach was better when mapping species that were less spectrally distinct by reducing the spectral variation within classes. The OBIA approach also reduced classification noise within the image. The study compared plants of both herbaceous and tree types and found that a height restriction rule imposed through use of a DSM could assist in the classification process. However, challenges such as radiometric inconsistency, poor spectral performance and DSM errors were encountered, and the authors commented with reference to data types and processing methods that the "(m)ethod of choice for a particular monitoring scenario thus depends on its purpose". Furthermore, they observed that although drone sourced imagery has advantages over satellite sourced imagery, it has limited spectral resolution and that the processing required for so many images renders it infeasible over a large area.

Various authors have investigated ways to reduce the workload involved in vegetation mapping, considering different approaches for automated detection. The combination of either PBIA or OBIA with a machine learning classifier is an established method for doing this. A recent study investigated the best approach for the mapping of *Harrisia pomanensis*, a cactus type plant (Mafanya *et al.*, 2017). To determine the best automated classification approach, supervised and unsupervised classifiers were used, which themselves varied between PBIA and OBIA approaches. Supervised classifiers (Maxver and Bhattacharya) performed better than unsupervised approaches (K-medians, Euclidian length and Isoseg), while the object-based Bhattacharya classifier performed better than the pixel-based Maxver one. Hence, the supervised OBIA approach was found to be best for mapping this species.

The OBIA approach has been the favoured one in recent works using high-resolution drone imagery, such as research by Chabot *et al.* (2018), Alvarez-Taboada *et al.* (2017), Martin *et al.* (2018) and Lehmann *et al.* (2017). Each of these authors used the OBIA approach for mapping invasive plants, of varying types and species. The methodology used by each of these authors is briefly described below.

Chabot *et al.* (2018) made use of multi-spectral imagery to map invasive water soldier (*Stratiotes aloides*) in waterways. Radiometrically calibrated imagery was "mosaicked and rendered into absolute reflectance maps (pixel values ranging from $0 - 1$) for each of the spectral bands" and a watershed algorithm was used to segment the image into objects. A set of features (14 spatial attributes, 4 spectral attributes, 4 texture attributes and entopy) were then extracted from these objects and used to train a random forest classifier. Chabot *et al.* (2018) noted that for features submerged underwater, that it was

best to collect imagery within two hours of the sun's zenith for best illumination.

An OBIA approach was also used by Lehmann *et al.* (2017) for analysing orthomosaics with the aim of mapping invasive *Acacia mangium* trees. Flights were performed in early morning hours to avoid strong wind and thermal activities three hours either side of noon, but the authors noted that partial cloud cover, low sun elevation angle, variations in illumination and haze may have affected the spectral integrity. Image equalisation was performed to reduce illumination differences. Vegetation indices were calculated, and the modified triangular vegetation index was found to be promising for segmenting the target plant. Semi-automatic classification was performed (unspecified classifier) using the classes *A. mangium*, grass, other vegetation, shadow, soil and road. A final interpretation map was produced which was the composite of the orthomosaic, vegetation index map, semi-automatic classification result and elevation map, which was useful since the target plant grew taller than other vegetation in the area. This was successful, but only partial tree crowns were detected, which was likely due to the low angle of the sun.

Another study made use of OBIA to map areas invaded by *Hakea sericea* (*Hakea*) in Portugal using multi-spectral imagery sourced from both satellite and drone (Alvarez-Taboada *et al.*, 2017). As is usual in remote sensing, an orthophoto was created for each image source, and features were extracted from objects within it and classified by a nearest neighbour classifier. A number of classes were used, namely *Hakea*, bare soil, woodlands, shrubs, infrastructure and unclassified. The drone-based sources achieved precision and recall above 75% (referred to as user and producer accuracies), with woodland often misclassified as *Hakea*. They noted that textural features, created using the local variance of three different pixel window sizes, did not improve accuracy within the *Hakea* class. The satellite data used in this study was sourced during the flowering season to increase the distinctiveness of the spectral signature, whereas the drone imagery was not.

Another study that made use of imagery from both satellites and drones was conducted by Martin *et al.* (2018). An OBIA approach was used to extract features from an orthomosaic with which to train a random forest classifier for the application of mapping Asian knotweed in two different landscapes (Martin *et al.*, 2018). As an additional feature, canopy height models were used. Knotweed obscured underneath tree-canopy was not visible to the camera and hence was not included. The authors found that the image source of choice depended on the landscape and spatial scale. They also used multi-date imagery and introduced a buffer boundary concept, which increased the accuracy for both satellite and drone imagery. Martin *et al.* (2018) noted that, when there is no distinctive phenological trait that can be exploited, increasing the number of variables used, such as

spectral channels or textural features, can improve accuracy.

Shiferaw *et al.* (2019) used hand-crafted features extracted from objects and compared the performance of different machine learning models for fractional cover mapping, particularly: gradient boosting machines, random forests, SVMs, deep neural networks, ensemble models and generalised linear models. Using this approach, random forests were found to perform the best. The authors noted that all models achieved better specificity than sensitivity. As is common in remote sensing applications, a distinctive phenological trait was used to assist in making the observed signal more distinctive. In this instance season was used, as the target plant was one of the few trees to retain leaves in the dry season against a backdrop of dry grass.

Göktogan *et al.* (2010) developed a system for detecting, classifying and spraying aquatic weeds. After acquisition of the aerial imagery, it was downloaded to the base station and processed into a mosaic. Small ROIs were selected and used as features for training an SVM. The trained SVM was used to output a weed probability distribution map, which was then used to guide which areas should be sprayed.

de Sá *et al.* (2018) investigated the use of drones for monitoring flowering invasive *Acacia longifolia* shrubs as a potential way of monitoring the effects of bio-control agents. RGB and colour infrared imagery were used to produce orthomosaics and canopy height models. One thousand points were used to train a random forest classifier. While the authors were able to show that it was possible to map the flowering of the weed accurately, they found that the flower counts from the drone imagery did not significantly correlate with those counted through fieldwork.

Baron *et al.* (2018) took a different approach, using PBIA, uncalibrated RGB imagery and single images as opposed to mosaics to identify invasive yellow flag iris (YFI) (*Iris pseudacorus*). Their previous work showed that orthomosaics could obscure target plants under tree canopy (Hill *et al.*, 2017). They investigated whether image pre-processing prior to classification by a random forest classifier would improve performance compared to the case when no pre-processing was used. Since their previous work showed that "manual digitisation provided more accurate maps than field surveys", YFI was marked with reference polygons through manual analysis of the imagery to produce a segmentation mask. The authors weighted their samples to produce a balanced dataset, and 68 features per pixel were computed, based on ten colour measures, four statistical features for each colour feature and texture features for non-HSV colour features. The use of ten features was found to be optimal with nine of these constant across the different experiments, namely " (1) mean HSV hue, (2) mean HSV saturation, (3) mean HSV

value, (4) standard deviation of HSV hue, (5) standard deviation of HSV saturation, (6) standard deviation of HSV value, (7) kurtosis of HSV saturation, (8) skew of HSV hue, and (9) skew of HSV saturation". The authors found that applying colour thresholding prior to classification reduced false positives as well as the computation time necessary for classification. Environmental factors such as lighting, viewing angle, differences in physical attributes and clustering of YFI, and vegetation maturity caused high variability in the characteristics of YFI blooms in the images sampled in this study. They note that their classifications suffered from a high false positive rate. De-speckling the image also improved classification by reducing false positives. They noted that the effect of image pre-processing, when combined with a supervised classification, improves overall accuracy by reducing false positives, at the expense of decreased sensitivity. They also mention that the number of images captured during an aerial survey is "overwhelming for manual image analysis" and that automated image classification, such as that shown in their work, will help scale this to larger areas.

Studies such as those mentioned thus far often make use of hand-crafted features, which are extracted from each of the objects. An alternative to this was demonstrated by Hung *et al.* (2014), who utilised a feature learning-based approach for the classification of three invasive weed species classes and a non-weed class in a UAV derived orthophoto. K-means clustering was used during training to develop a set of image filters, which were pooled for each class. A class was then defined by one or more centroids of the clusters (called textons). The resulting histograms for each class were visually distinct, indicating that class separability was possible. Altitudes varying from 5 to 30 m were tested, with images in the 5 to 10 m range found to deliver the best performance. The resulting system was also shown to give strong F1 scores.

Deep learning algorithms allow for the learning of more robust features. Within the broader remote sensing field, applications have been adopted in all stages of remote sensing data analysis (Zhang *et al.*, 2016). Furthermore, Mboga *et al.* (2018) compared an FCN to a state-of-the-art objected-based approach to city structure classification and found that the FCN was better able to generalise than the OBIA approach as it learnt features directly from the image. Carrio *et al.* (2017) noted that deep learning-based feature extractors are often based on CNNs and are used for object recognition and scene classification.

Within the plant detection and mapping space, Liu *et al.* (2018) contends that FCNs and patch-based deep CNNs give better accuracies than random forest and SVMs for OBIA based classification for the task of mapping wetlands. However, a caveat is that this is true only with large training sample sizes.

A comparative study of deep learning and state-of-the-art OBIA techniques for the detection of scattered shrubs was performed by Guirado *et al.* (2017). They made use of two classes, invasive *Ziziphus lotus* and "bare soil with sparse vegetation". Two CNNs were investigated, namely ResNet and GoogLeNet. These were investigated under fine-tuning (with weights from ImageNet) and fine-tuning with data augmentation optimisations. The approaches of using a sliding window and object proposals were both investigated. The OBIA methods first made use of a segmentation algorithm using features such as scale, shape, colour, compactness and smoothness. Thereafter, KNN, random forest and SVM classifiers were applied. It was found that the best performing CNN out-performed the best performing OBIA technique, improving upon precision, recall and F1 score. This was the ResNet-based classifier, trained with both fine-tuning and data augmentation and applied using object proposals. Additionally, the CNN "required less human supervision than OBIA", was trained "using a relatively small number of samples" and "the detection process is faster with the CNN-detector than with OBIA". OBIA methods are also tuned for a particular image and are not as transferable as CNNs. The authors note that incorporating CNNs as classifiers into the OBIA approach could leverage the strengths of each.

The use of CNNs to learn texture features for semantic segmentation has been demonstrated by Yao *et al.* (2016) on urban aerial imagery. To the best of our knowledge, no published literature is yet available on the use of deep learning-based semantic segmentation in an end-to-end fashion for the mapping of invasive plants. Chabot *et al.* (2018) observed that traditional machine learning techniques are constrained both by how objects are segmented and by the set of hand-chosen features selected. An end-to-end deep learning semantic segmentation approach, such as that used by Yao *et al.* (2016), could address this problem and compete with state-of-the-art OBIA based methods.

3.3 Summary

A general overview of drone applications in conservation and research was presented in this chapter, along with discussions on the techniques used for detecting objects in aerial imagery in similar fields. The benefits of drones, both as a research tool and for use in conservation related tasks, is evident from the many applications discussed. The use of learnt features shows promise in many areas. In the following chapter, the research design is introduced, giving detail on its approach to meeting the research objectives.

Chapter 4

Research Design

In this chapter, the approach to this research is introduced and justified. An overview of the research design is given, followed by an outline of the hardware and software used, the particular invasive plant chosen as a target case study and contextualisation of the location of the study. This chapter also covers the development of relevant datasets for supervised classification and their annotation.

4.1 High-level research design

The aims of this research require that various machine learning models are assessed, an optimal one is selected, trained and integrated with a drone system and then tested in the field under real conditions. CNNs were selected, due to their known performance on image classification challenges, as the family of models to investigate. Particularly, we selected a set of well-known models for classification (Xception, Inception, MobileNet and SqueezeNet) and an FCN architecture called U-Net for semantic segmentation.

A case study approach is taken, with a single invasive shrub species chosen as the target shrub and a relevant dataset collected using the available drone. The collected dataset consists of single images, rather than orthomosaics as were often seen in the literature, containing a mixture of both *Hakea* and other shrubs in amongst sparse vegetation. This was a suitable choice as the aim of the project was not to map an area, showing a population count, but rather to detect target shrubs on the fly.

The two machine learning approaches, candidate proposal followed by classification and semantic segmentation, were chosen because although they both offer a possible solution

to the task, they offer different advantages and disadvantages. Object detection networks, such as YOLO and R-CNN, were not included in this study as the bushes grow both in a scattered formation and in dense stands, where individual instances of the target shrubs are difficult to differentiate even to the annotator.

The classification approach illustrates the accuracy attainable when presented with the ideal case – close-cropped windows containing a shrub of either the target or other class. The viability of this approach was then investigated further, in terms of what measure of success can be achieved through different candidate window proposal strategies. A second approach, deep learning-based semantic image segmentation, demonstrates a different strategy with end-to-end training for pixel-wise classification.

From a discussion of the advantages and disadvantages of these two approaches, a model can be selected and integrated with the drone system for preliminary field testing. This application should provide visual feedback when a shrub of the target class is detected. Preliminary results give insight into model shortcomings, which are then addressed through model refinement, particularly the use of further relevant data augmentation and loss functions. The final model is then evaluated in the field. A visual representation of the research design used in this thesis is shown in Figure 4.1.

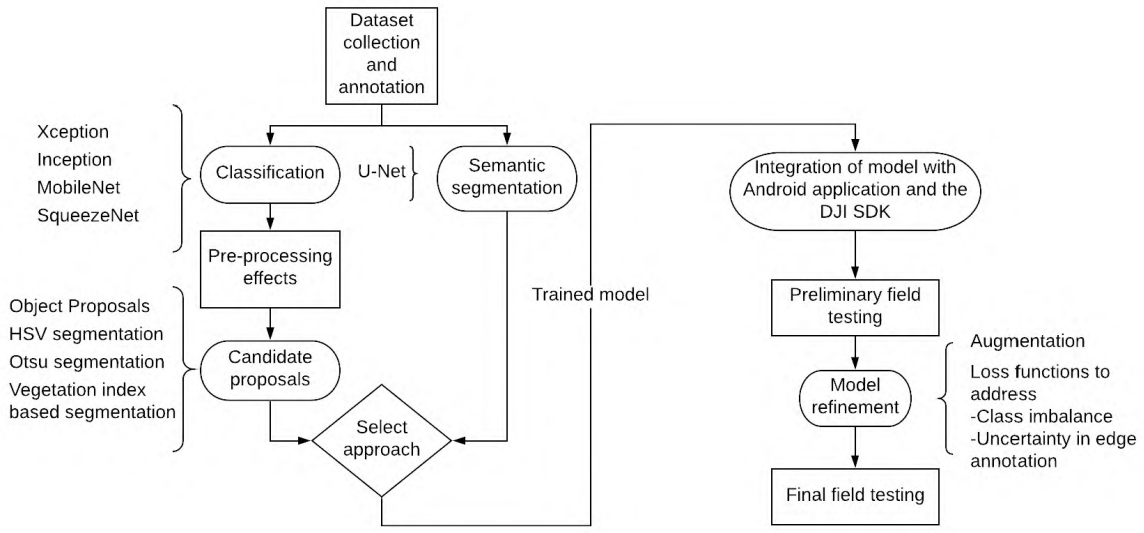


Figure 4.1: Visual overview of research design.

4.2 System infrastructure

Various hardware and software components were necessary for this research. This section covers the selection of suitable components, indicating requirements and specifications.

4.2.1 Hardware

A drone platform suitable for the detection of alien vegetation is necessary. Drone candidates are required to be quadcopters, as hovering ability is needed such that the drone is able to remain stationary above an object of interest to indicate whether it is a target or not. Hovering ability also allows for expansion to a herbicide/bio-control delivery system where it would be necessary to drop a payload on the target plant. It is also imperative that the drone has a reasonable battery life, as the areas to be surveyed can be extensive. A camera capable of pointing vertically downward is needed to obtain aerial imagery upon which target detection can be performed. Most importantly, the drone must be programmable such that it can interact with the trained model.

The DJI Mavic Pro[1] meets these requirements, with 4K/30fps video, a 12 MP camera, GPS/GLONASS, 27 minutes of battery life, flight distance from the ground station up to 7 km and a mobile SDK. The live view video is accessible at 720p or 1080p at 30 fps via the remote controller, with a latency of 160-170 ms.

The chosen groundstation is an Asus ZenPad3s10[2]. This is a powerful tablet, with a hexa-core and 4 GB RAM. The hardware setup is shown in Figure 4.2.

4.2.2 Software

Python[3] provides good support for machine learning, supporting platforms such as *Scikit-Learn*[4], *Caffe*[5], *TensorFlow*[6] and *Theano*[7]. As such, it was chosen as the language in

[1] https://www.dji.com/mavic/info#specs
[2] https://www.asus.com/us/Tablets/ASUS-ZenPad-3S-10-Z500M/
[3] https://www.python.org/
[4] https://scikit-learn.org/stable/
[5] https://caffe.berkeleyvision.org/
[6] https://www.tensorflow.org/
[7] http://deeplearning.net/software/theano/

Figure 4.2: Chosen hardware and setup, the DJI Mavic Pro and Asus ZenPad3s10.

which to develop and train the machine learning models described in this thesis. *TensorFlow* was chosen as the machine learning platform, with the *Keras*[8] high-level application programming interface (API) used to allow for easy prototyping. Furthermore, *TensorFlow* supports both Python and C and is backwards compatible with C++, Go, Java, JavaScript, and Swift. This is an important factor for selection, as the developed model is required to be integrated with the DJI SDK in an Android Application, which is Java based.

4.3 Target shrub genus

According to Richardson *et al.* (1987), four species of *Hakea* were introduced to South Africa in the mid-1800s as a hedge plant from Australia. Richardson *et al.* (1987) state that of these four species, three are invasive, and of these, *Hakea sericea* is highly invasive and the most prevelent of all woody invaders in South Africa's fynbos biome (Figure 4.3). Unlike indigenous members of the *Protacea* family, shrubs in the *Hakea* genus do not suffer from seed predation, resulting in the germination of large seed banks after fire and the formation of dense stands, as seen in Figure 4.3(b), which alter the composition of the ecosystem and crowd out indigenous fynbos (Richardson *et al.*, 1987). Furthermore, the authors note that they are far heavier consumers of water than the indigenous fynbos with which it competes, which also reduces run-off. *Hakea sericea* is also invasive in Portugal

[8] `https://keras.io`

(Alvarez-Taboada *et al.*, 2017) and New Zealand (Kluge & De Beer, 1984).

(a) Side profile of a *Hakea* shrub

(b) Unchecked invasion results in dense stands

Figure 4.3: Photographs showing the target shrub genus, *Hakea*.

In addition to its negative environmental impact, this genus of shrub was selected as the target genus for this project due the ease of accessibility to invaded land as well as the researcher's familiarity with the invasive shrub, which is necessary for annotation of the images. Shrubs of this genus are referred to as 'target shrubs' in this thesis, unless it is appropriate to mention the genus name.

(a) Aerial view of a flowering *Hakea*

(b) Invaded slope of flowering *Hakea*

Figure 4.4: *Hakea* in its flowering season.

The *Hakea* genus flowers in late winter (June to September), when it is covered in cream blooms (Kluge & De Beer, 1984), such as illustrated in Figure 4.4. This is a distinct phenological stage in which the shrub looks different to other shrubs, but unlike the majority of studies in the literature, the dataset was collected when the shrub was not in its most distinct phenological stage. This is because the flowering season is not particularly long, and a useful tool would require detection throughout the remainder of the year.

4.4 Study site

Imagery of the target shrub was collected from a farm south-west of Grahamstown, in the Eastern Cape, South Africa. The site falls within the hilly section of Figure 4.5 (a), at roughly -33.366 S, 26.516 E, where two species of *Hakea* have been documented to have invaded, namely *Hakea suaveolens* and *Hakea sericea* (Palmer, 2004). Written permission to fly the drone and collect imagery for the dataset was obtained from the relevant landowner of the farm, and is included in Appendix A. Early experimentation with data collection was also performed at a second farm, but due to a fire that destroyed the shrubs at the intended collection site, this farm was not included in the study.

(a) Location of Grahamstown within South Africa. Image reproduced from Google Maps (Map data: AfriGIS (Pty) Ltd, 2019)

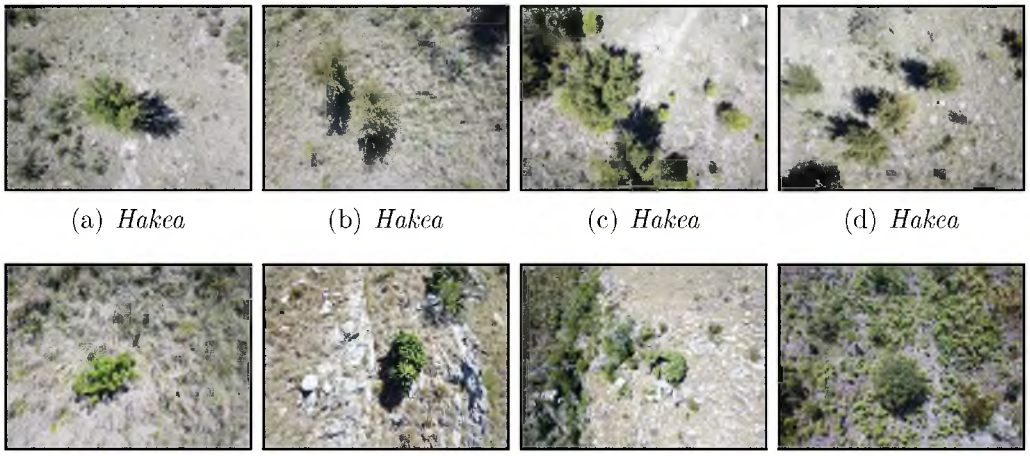
(b) The location of the data collection site relative to Grahamstown. The site is marked by a red location marker. Image reproduced from Google Maps (Imagery: CNES/Airbus, Digital Globe, 2019. Map data: AfriGIS (Pty) Ltd, 2019).

Figure 4.5: Location of study site.

4.5 Collected datasets

Two image datasets were recorded: dataset 1 and dataset 2, which were collected on different days, approximately a month apart but both during summer. The drone was

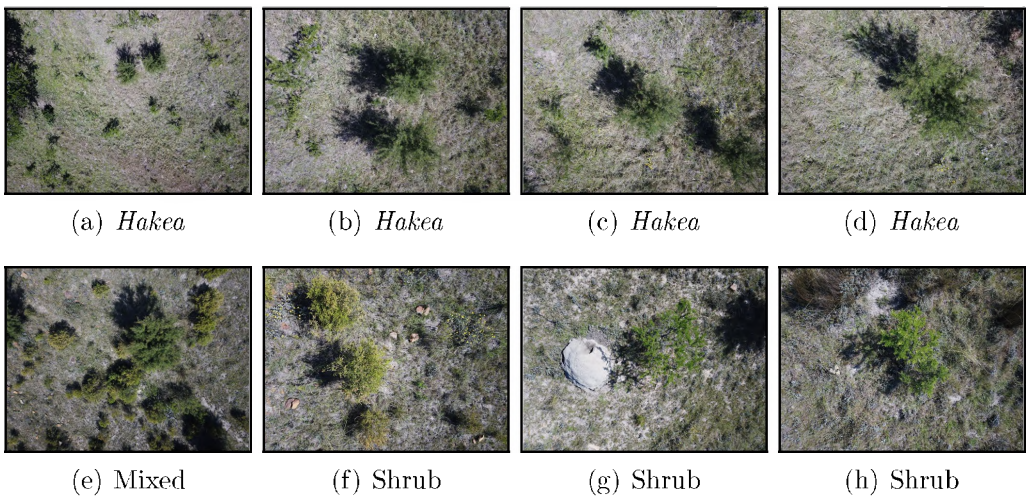
flown with its camera pointing vertically down to hover above a series of target and non-target shrubs. Flights were conducted between 10 AM and 3 PM to minimise the shadows of shrubs and care was taken to avoid placement of the drone's shadow on the bush. The target shrub was positioned such that the entire shrub of interest fell within the field of view and a photograph was taken. Manual camera settings were adjusted based on the light intensity of the day such that the exposure appeared balanced on the livestream feed. Images of the target shrub were taken at different heights so as to allow for variation in size of the shrub within the image, as in the hilly to mountainous terrain in which *Hakea* occur it is difficult to ensure the drone flies at a constant height above ground. Sizes of plants also vary, so data with varying altitudes also assist in developing a classifier that is invariant to size.



(a) *Hakea* (b) *Hakea* (c) *Hakea* (d) *Hakea*

(e) Shrub (f) Shrub (g) Shrub (h) Shrub

Figure 4.6: A sample of dataset 1.



(a) *Hakea* (b) *Hakea* (c) *Hakea* (d) *Hakea*

(e) Mixed (f) Shrub (g) Shrub (h) Shrub

Figure 4.7: A sample of dataset 2.

The resulting datasets contain images such as those shown in the following samples, dataset 1 in Figure 4.6 and dataset 2 in Figure 4.7. There are 120 images in dataset 1, of which 52 images contain at least one target shrub and 82 images in dataset 2, of which 48 contain at least one target shrub.

4.6 Annotation of datasets

Each frame within the dataset was annotated in two ways, the first suitable for a classification task and the second for a semantic segmentation class. For the classification task, simple bounding boxes were placed around all target shrubs using LabelImg, a simple graphical image annotation tool for placement of bounding boxes, as shown in Figure 4.8. Coordinates of the bounding boxes were saved to a .txt file in YOLO format. This allowed the extraction of close-cropped sub-images containing only the shrub of interest and made use of the saved coordinates and class label produced by LabelImg.

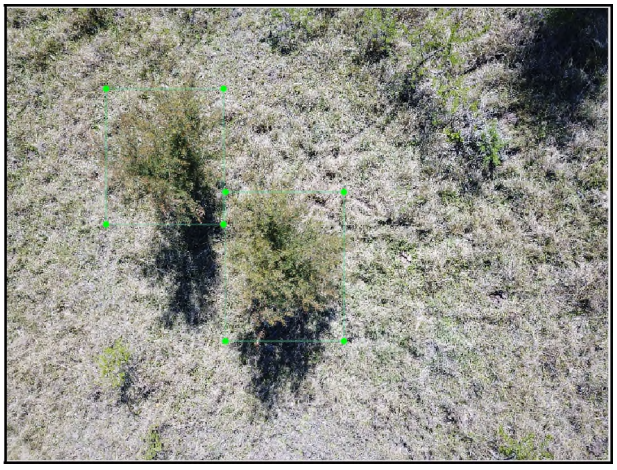


Figure 4.8: An example of annotation using LabelImg.

The sub-images were extracted at a constant aspect ratio of 1:1, as the majority of classifiers expect the input dimensions to be the same. Extraction at constant aspect ratio ensures that resizing the image would not cause distortion and hence introduce an unintentional bias in the model. A code snippet for the adjustment of the aspect ratio is presented in Listing 4.1, showing how the aspect ratios were adjusted to 1:1 by increasing the smaller of the two dimensions and discarding sub-images in which this led to the bounding box exceeding the dimensions of the image.

A sample of the extracted images is presented in Figure 4.9. The effect of increasing the smaller dimension of the sub-images is clear in Figure 4.9 (a) and (g), where there is a clear

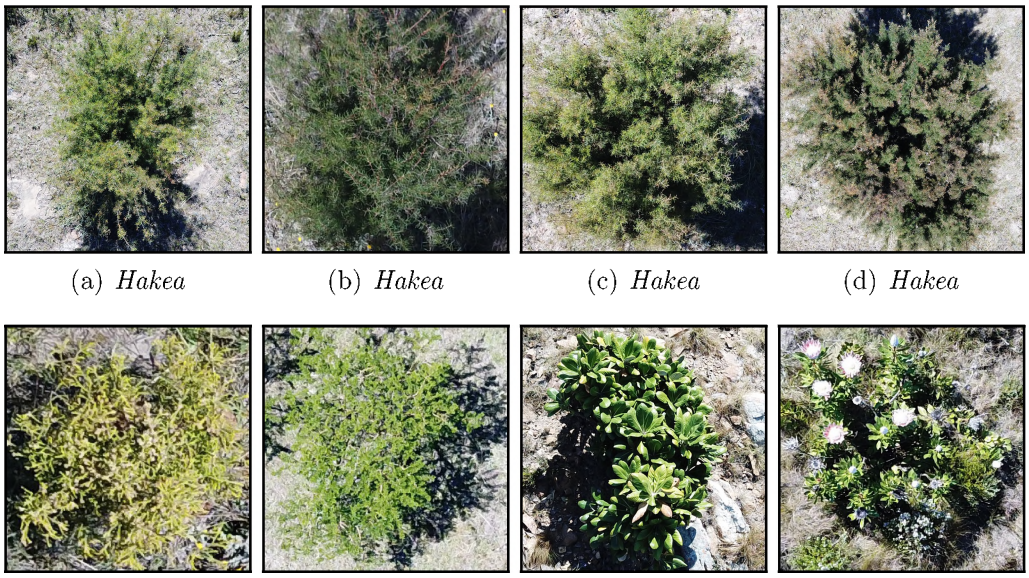
```
for l in lines:
        c,x,y,width,height = l.split(" ")
        x,y,width,height = float(x),float(y),float(width),
    float(height)
        x,y,width,height = int(x*w),int(y*h),int(width*w),
    int(height*h)
        #constant aspect ratio (square)
        if(width > height):
            d = width-height
            height += d
        else:
            d = height-width
            width += d

        x,y = int(x-width/2.0),int(y-height/2.0)
        if(x < 0 or x + width >= w or y < 0 or y >= h):
                continue #discard where subim exceeds side

        subim = img[y:y+height,x:x+width] #extract subimage
```

Listing 4.1: Adjustment of aspect ratio.

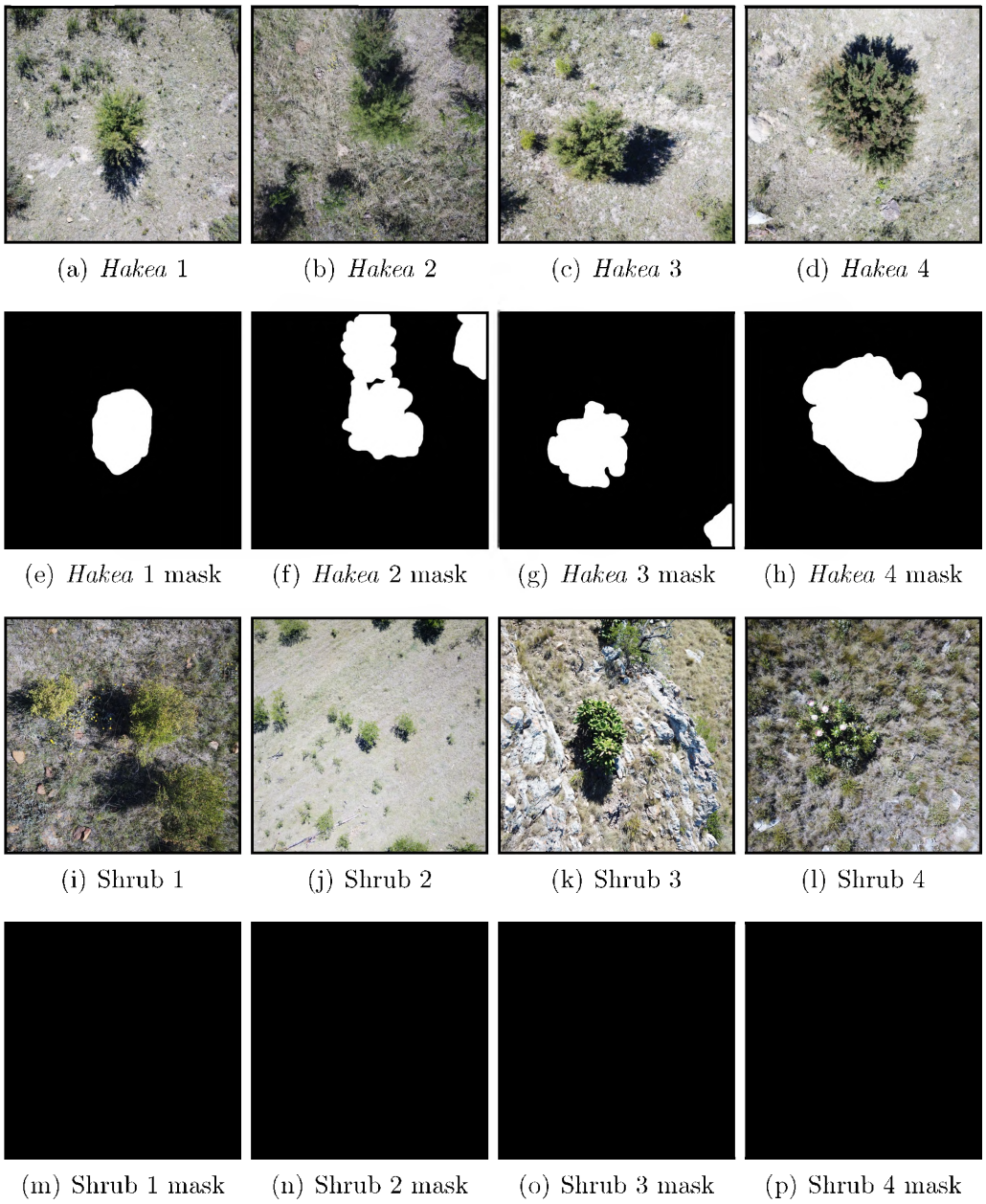
gap between the shrub and the edge of the image. After discarding sub-images in which the shrub could not be extracted at constant aspect ratio, a total of 147 close-cropped images of target shrubs and 133 of other shrubs in dataset 1 were extracted, while in dataset 2, 99 boxes containing target shrubs and 107 of other shrubs were extracted.



(a) *Hakea* (b) *Hakea* (c) *Hakea* (d) *Hakea*

(e) Shrub (f) Shrub (g) Shrub (h) Shrub

Figure 4.9: A sample of the extracted images used for the classification task.



(a) *Hakea* 1 (b) *Hakea* 2 (c) *Hakea* 3 (d) *Hakea* 4

(e) *Hakea* 1 mask (f) *Hakea* 2 mask (g) *Hakea* 3 mask (h) *Hakea* 4 mask

(i) Shrub 1 (j) Shrub 2 (k) Shrub 3 (l) Shrub 4

(m) Shrub 1 mask (n) Shrub 2 mask (o) Shrub 3 mask (p) Shrub 4 mask

Figure 4.10: A sample of the extracted images used for the segmentation task.

The second annotation was a pixel-level semantic annotation to produce ground truth segmentation masks using the GNU Image Manipulation Program[9] (GIMP), in which portions of the image containing target shrubs were masked by hand as white, while all other areas were masked with black. Prior to masking in GIMP, images were cropped to a 1:1 aspect ratio, as required for the image segmentation architecture, discussed in Section 5.5.1. Samples of the semantically annotated dataset are shown in Figure 4.10.

[9]`https://www.gimp.org/`

Unlike in the case of annotation for classification, semantic annotation masks include all instances of the target shrub, including those at the edges of the image. When considering the entire dataset, only a small proportion of the pixels are of the target class, indicating class imbalance and the necessity for using appropriate metrics to evaluate model performance.

4.7 Summary

In this chapter, a high-level overview of the research design was presented and the hardware and software necessary to approach this were introduced. Furthermore, two datasets were collected to assist in the training and testing of the models developed in the following chapters. The first dataset was collected with the intention for use as training data, while the second was collected for testing to ensure generalisation of the model to differing environmental conditions and unseen data.

Chapter 5

Model Design and Implementation

In the previous chapter, we introduced the collected datasets of images of the target shrub, *Hakea*, and other shrubs. In this chapter, we make use of these images in training various machine learning architectures as a step towards answering our research question by developing a model capable of detecting shrubs of the *Hakea* genus. This model forms the intelligent part of the system for field testing with the drone. As such, this chapter proposes to determine whether a neural network-based model can be trained such that it can reliably detect target shrubs within a frame.

In this chapter, we examine different architectures and approaches to developing a model for *Hakea* detection. Particularly, we make use of fine-tuning to train existing classification architectures on our dataset and assess algorithms for proposing segments of the image for classification. We then investigate image segmentation as an alternative to separate proposal and classification and compare the performance of the two methodologies. This allows us to make a recommendation for the best model for *Hakea* detection, which is crucial to developing a robust system for deployment in the field testing system.

In Section 5.1, we give an overview of the related work specific to this chapter. Thereafter in Section 5.2, we obtain a best estimate of the possible accuracy achievable on our dataset when using well-known existing architectures and a fine-tuning procedure. Section 5.3 follows to determine whether a pre-processing function can improve on the accuracy obtained as a baseline in the previous section. An examination of different candidate proposal algorithms is presented in Section 5.4, which are necessary to propose segments of the larger frame for the classifier to be applied to as a subset of the larger frame. An alternative approach to the separate stages of proposal and classification is investigated in Section 5.5, in which we examine the possibility of using a segmentation algorithm to

classify the individual pixels within the image into the two classes. Finally, we present our findings and make a recommendation based upon them.

5.1 Related studies on deep learning in plant identification

Multiple approaches to plant classification are present in the literature. According to the review by Wäldchen *et al.* (2018), there has been a significant amount of work done at species level using single organs of plants, such as leaves, flowers, stems or fruit. They noted that several benchmark datasets had been developed for this, such as Swedish leaf, Flavia, Leafsnap and ICL. These datasets vary from having 15 to 1000 species and from datasets of 1125 images to 113205 images, which means between 75 and 113 images per class if balanced. The review goes on to say that multi-image algorithms have the potential to create a more robust classifier than those based on a single organ, making use of more than one view so as to assess the different organs best. When comparing the accuracy of deep learning classifiers against that of model-based and model-free methods, such as the scale-invariant feature transform, the review found overall that deep learning methods achieved substantially better performance.

The datasets discussed above require that the leaf or organ is placed against a plain background when the image is captured. As noted in their review, these laboratory conditions make the classification problem significantly simpler than images taken in their natural environment, which often have complex backgrounds that increase the difficulty of segmenting the image. With the aim of more widespread botanical surveys conducted using automated classification applications, awareness of the importance of classification in the natural scene has become apparent and several datasets have been created to address this, namely Oxford Flower 17, Jena Flower 30, Oxford Flower 102 and PlantCLEF16, which have natural backgrounds (Wäldchen *et al.*, 2018).

Wäldchen *et al.* (2018) also mention several classifiers trained on these benchmarks which made use of transfer learning, such as an AlexNet classifier, with weights pre-trained on the ImageNet dataset, which was fine-tuned to classify 44 plant species using their leaves and achieved 99.5% accuracy. Similarly, a six-layer CNN was used to classify the leaves in the Flavia dataset, consisting of 33 classes, and achieving 94.69%, while a 17 layer version later achieved 97.9% accuracy on the same dataset. This was then outperformed by a 26 layer ResNet architecture which achieved 97.9% accuracy.

Beyond acting as a classifier, CNNs can be used as a feature extractor. An example of this within the plant identification literature, as reported by Wäldchen *et al.* (2018), was the use of the features extracted by a CNN as input to an SVM for classification of the Oxford Flowers 102 dataset, which achieved 95.34% accuracy. These benchmark datasets are useful, but in cases where the plant is photographed in laboratory conditions Wäldchen *et al.* (2018) caution that the accuracy obtained should be viewed as a gauge of best possible performance, as in the real world backgrounds and environmental conditions are not controlled.

Several mobile applications have also been developed for plant identification. Particularly notable are LeafSnap (Kumar *et al.*, 2012) and Pl@ntNet (Goëau *et al.*, 2013). LeafSnap makes use of images taken on a mobile phone to identify tree species using images of their leaves and was the first mobile application to make use of images for plant identification. According to Kumar *et al.* (2012), this system required that leaves be placed on a solid, pale background. The system consisted of an SVM with RBF kernel to first distinguish between leaf and non-leaf images. Images classified as leaves were segmented using the HSV colour space before a curvature histogram was extracted and this was fed to a nearest neighbour classifier. The study also states that LeafSnap was trained on images of leaves from 184 species of tree and its dataset consisted of 23915 images from laboratory conditions and 5192 images taken by mobile devices in the field. The application achieved 96.8% top-5 accuracy.

Unlike LeafSnap, Pl@ntNet made use of multiple organs, namely flowers, leaves, fruit and bark, to identify plants (Affouard *et al.*, 2017). Initially, Pl@ntNet made use of hand-crafted features, but in recent years has shifted to the use of a CNN. At the time of writing, the classification model used the Inception model extended with batch normalisation and had been trained on 10 000 species and 332 000 images (Affouard *et al.*, 2017). Multiple images of the same plant can be used to create a prediction, using a weighted average of the softmax outputs from the classifier. Other plant identification applications exist, such as MedLeaf (Prasvita & Herdiyeni, 2013) for the identification of medicinal plants using leaf images. This network used local binary patterns for texture extraction and fed the output of this to a probabilistic neural network. Applications for plant species identification typically focus on broad-leafed plant species, with fine leaves and needles generally excluded (Winberg *et al.*, 2017).

An exception to this trend is the application FLORA, which was developed to identify fynbos leaves, which are typically small, fine or needle-like in shape (Winberg *et al.*, 2017). This application made use of a KNN approach to classification and achieved a

top accuracy of 87%. While this accuracy is distinctly high, it is interesting to note the difference in performance when compared to the over 90% accuracies achieved for the broad-leaf classifiers discussed previously.

The remote sensing community has long used natural scene imagery from satellites and manned flights to map vegetation types, although frequently making use of multi-spectral data to assist in this process. Despite the use of multi-spectral imagery, the use of machine learning to identify the species of interest still indicates a high relevance of the literature in this area. In remotely sensed imagery, the imagery is taken from a vertical viewpoint, so classifications are made using the vertical view of the whole plant, rather than through the use of a single organ. Random forests and SVMs are well established in this field, but it has been found by Liu *et al.* (2018) that if sufficient training data is available, then CNNs can outperform SVMs and random forest classifiers, as demonstrated when applied to wetland mapping. Deep learning has also begun to be present, to varying degrees, in all stages of remote sensing analysis, namely pre-processing, pixel-based analysis, target recognition, feature extraction and scene understanding (Zhang *et al.*, 2016).

In the remote sensing literature, there are many examples of work in which deep learning is being applied to plant identification, although there are still many more in which random forests are applied. A sample of these papers is mentioned here. Guirado *et al.* (2017) compared the use of OBIA and deep learning methods for distinguishing *Ziziphus lotus* shrubs from bare ground with sparse vegetation and found that the deep learning methods had better performance. The deep learning architectures used in this research were GoogLeNet (which is based on the Inception model) and ResNet. The classifiers were evaluated through both a sliding window approach and an object proposal approach based on colour segmentation and pixel clustering. It was found that ResNet was the best performing classifier when coupled with the object proposal technique with 100% precision, 93.24% recall and 96.50% F1 score. Only 100 images per class were used, but through data augmentation 6000 were obtained.

GoogLeNet has also shown success in other applications, such as classifying seven tree types at 89.0% accuracy from high-resolution imagery, captured by a drone (Onishi & Ise, 2018). There have been several other examples of deep learning applied to drone sensed imagery to detect plants. In such agricultural applications, Carrio *et al.* (2017) state that these systems are usually used for surveys in which plants are classified and counted. Architectures used for this task include AlexNet and a hybrid Feedforward Neural Network-CNN.

Across the board, it appears that machine learning is replacing traditional methods of

plant classification, due to the high accuracy achievable through the ability of deep learning networks to learn features, rather than relying on hand-crafted ones. Successful attempts have been made to classify plants based on imagery of just one organ, multiple organs or a vertical view of the whole plant as in remote sensing. Training sample sizes also vary across the literature, but it is noted that relatively small datasets can be used to produce high accuracy, particularly when used in conjunction with a pre-trained network in a fine-tuning process.

5.2 Application of existing classifier architectures to research dataset

In this section, we investigate the performance of existing, pre-trained models when fine-tuned using the collected dataset. The performance of these models yields a baseline accuracy for the classification of the image dataset into the *Hakea* and other shrub classes and allows for future comparisons. For simplicity, from this point, the *Hakea* class will be referred to as the 'target class' and the class containing other shrubs will be referred to as the 'other' class.

To determine a reliable estimate of performance, fine-tuning was performed using k-fold cross-validation and mean metrics were calculated. The models were then fine-tuned using the entirety of dataset 1 as training data, with no validation data, with the number of epochs chosen using the results from the k-fold validation as a guide. These models were suitable for inference on dataset 2, which allowed metrics to be calculated to ensure the models were capable of generalising to unseen data. Thereafter, class activation maps were created for each of the architectures. After considering the k-fold results and class activation maps, a final model was selected to provide a baseline accuracy to be used for inference in Sections 5.3 and 5.4.

5.2.1 Transfer learning model selection

Keras offers a number of models, available with weights, that were pre-trained on the ImageNet dataset, as listed in Table 5.1. To establish a baseline accuracy for classification of the target and other classes, a selection of these pre-trained models were fine-tuned using the collected dataset.

Table 5.1: Models pre-trained on ImageNet dataset that are available in Keras

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88	0.790	0.945	22 910 480	126
VGG16	528	0.713	0.901	138 357 544	23
VGG19	549	0.713	0.900	143 677 240	26
ResNet50	99	0.749	0.921	25 636 712	168
InceptionV3	92	0.779	0.937	23 851 784	159
InceptionResNetV2	215	0.803	0.953	55 873 736	572
MobileNet	16	0.704	0.895	4 253 864	88
MobileNetV2	14	0.713	0.901	3 538 984	88
DenseNet121	33	0.750	0.923	8 062 504	121
DenseNet169	57	0.762	0.932	14 307 880	169
DenseNet201	80	0.773	0.936	20 242 984	201
NASNetMobile	23	0.744	0.919	5 326 716	-
NASNetLarge	343	0.825	0.960	88 949 818	-

Models with a size under 100 MB were considered as candidates for the transfer learning model. The Xception and InceptionV3 models were chosen from this selection as they had the highest top-5 and top-1 accuracies. MobileNet was also selected for comparison, as it has a significantly smaller model size than either the Xception or Inception models. Finally, the SqueezeNet model, from the *keras_ squeezenet* module[1] was also selected due to its small model size. Hence we have two models selected for accuracy (with acceptable model size) and two models selected for model size.

Each of these models has the option to exclude the final layers of the model, called 'top-less', so that the model may be used as a feature extractor and then fine-tuned through the addition of a few untrained layers for classification. Each of the models was loaded in top-less mode to produce a base model. All the layers in the base model were frozen so that they would not be affected by the fine-tuning process. A global average pooling layer, dense layer with 1024 units and relu activation and final dense layer with softmax activation were added to the base model for training to distinguish between our target and other classes. This is a popular configuration for fine-tuning, recommended in the Keras documentation.

5.2.2 K-fold cross-validation

Ten-fold cross-validation was performed on each of the models, which were trained using an early stopping criterion to prevent over-fitting. Dataset 1 was used to fine-tune the models and was split into ten folds using Scikit Learn's `StratifiedKFold`, which keeps

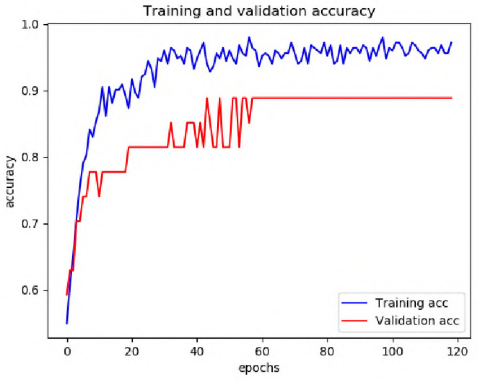
[1] `https://github.com/rcmalli/keras-squeezenet`

the percentage of images in each class constant across the folds, reserving one fold at a time for validation. To improve the performance of the model, data augmentation was performed on the training set using the parameters in Listing 5.1. The ImageDataGenerator function made use of these parameters to produce additional augmented images from the flow of images during the training process. Figure 5.1 shows the training graphs for the tenth fold of each model, while the mean metrics computed across the ten folds are presented in Table 5.2.

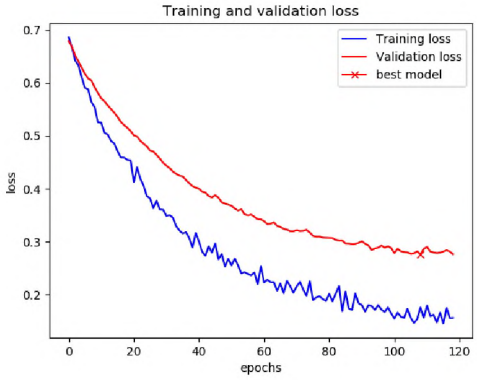
```
data_gen_args = dict(horizontal_flip=True,
                     rescale = 1./255,
                     vertical_flip=True,
                     rotation_range=20,
                     width_shift_range=0.2,
                     height_shift_range=0.2,
                     zoom_range=0.2,
                     fill_mode='nearest',
                     preprocessing_function=None
                     )
```

Listing 5.1: ImageDataGen parameters.

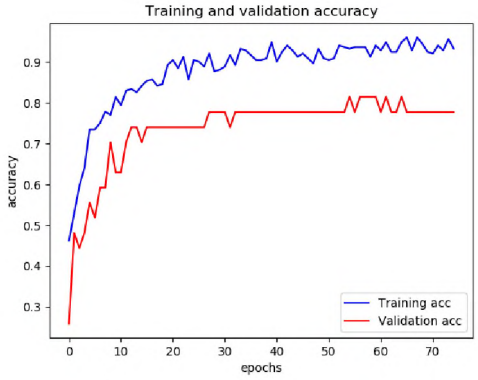
From the results in Table 5.2, it is seen that good accuracies were achieved for all models, with the Xception model the best performer, closely followed by MobileNet and Inception. This ranking holds true for the F1-score as well, which indicates a good balance between precision and recall. The Inception and MobileNet models achieved perfect and near-perfect precision respectively, which indicates that the false positive rate is virtually non-existent, so when a positive is predicted, it is almost certain to be a ground truth positive. However, these models achieve such high precision at a cost to their recall score, as both achieve values in the mid to low seventies. This indicates that because the models are so conservative about predicting an input as part of the target class, approximately 25 to 30 percent of the ground truth positives are not predicted as positives. In contrast, the Xception and SqueezeNet model show the opposite behaviour, with high recall and lower precision, indicating that the models classify the vast majority of ground truth positives as positive but at the cost of an increased false positive rate. The Xception model does the best job of balancing this trade-off, as is evident by its high F1-score, with a recall of 0.97 and a precision of 0.87, indicating that only 3% of ground truth positives are classified as false negatives and 13% of predicted positives are false positives. The choice of preferable behaviour depends on the application of the model. If this involved counting target shrubs, there should be no false negatives (target shrubs classified as shrubs in



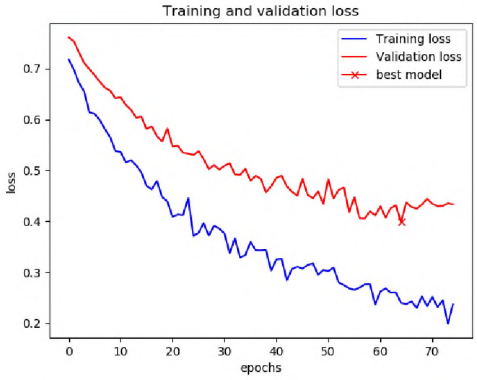
(a) Xception accuracy



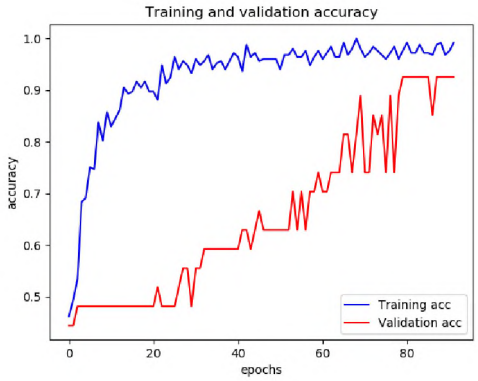
(b) Xception loss



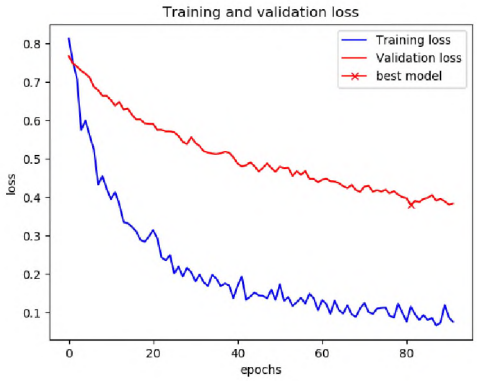
(c) Inception accuracy



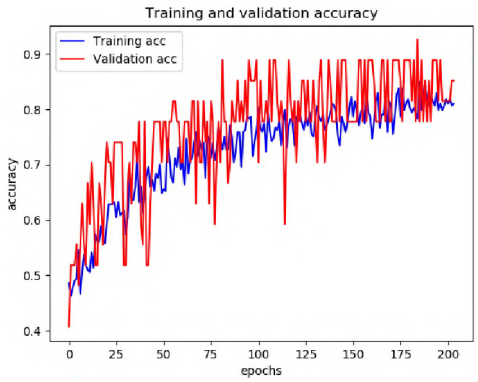
(d) Inception loss



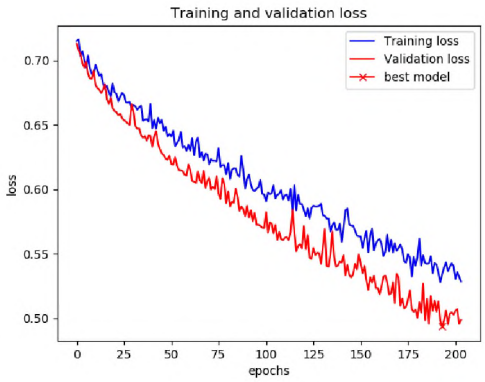
(e) MobileNet accuracy



(f) MobileNet loss



(g) SqueezeNet accuracy



(h) SqueezeNet loss

Figure 5.1: Learning curves for models produced by the final fold of 10-fold cross-validation.

the "other" class). Since further identification of candidate *Hakea* could be utilised and any non-*Hakea* removed from the list manually, the Xception model would be preferable for this task. However, if the model were to be used in a precision spraying application, false negatives would be preferable to false positives, as indigenous vegetation should not be sprayed. In this instance, the performance of the MobileNet model would be preferable.

Table 5.2: Mean metrics with standard deviations calculated from 10-fold cross-validation for each model

Model	Accuracy	Precision	Recall	F1 score
Xception	0.90 ± 0.07	0.87 ± 0.09	0.97 ± 0.05	0.91 ± 0.06
Inception	0.85 ± 0.06	1.00 ± 0.00	0.71 ± 0.11	0.83 ± 0.08
MobileNet	0.87 ± 0.07	0.99 ± 0.02	0.75 ± 0.14	0.85 ± 0.10
SqueezeNet	0.75 ± 0.13	0.70 ± 0.08	0.99 ± 0.03	0.81 ± 0.08

The variance across all ten folds was calculated to give an indication of how the metrics fluctuated across the folds. A low variance is preferable to a higher one, as high variance indicates that the model may have over-fit to the training data. This can occur either through the use of a model that is too simplistic for the data or if there is insufficient data to train the model to convergence. The variance in the accuracy metric and F1-score indicates that the Xception and MobileNet models are both reasonably reliable, although the MobileNet model shows a higher variance on its recall score.

5.2.3 Inference models

Each of the models was fine-tuned, with the number of epochs for Xception, Inception, Mobilenet and Squeezenet set to 150, 50, 100 and 180, respectively. These values were guided by observing the general training period during the k-fold validation. The models were each trained using the entirety of dataset 1, with no samples excluded for validation. Once trained, the models were then evaluated on dataset 2, which was unseen to the trained models. The results of this analysis are shown in Table 5.3.

Table 5.3: Performance of each model on the unseen test set, dataset 2

Model	Accuracy	Precision	Recall	F1 score
Xception	0.83	0.75	0.98	0.85
Inception	0.91	0.95	0.85	0.90
MobileNet	0.98	1.00	0.95	0.97
SqueezeNet	0.72	0.63	1.00	0.78

When comparing the performance of these models on the unseen test set, to that of the average validation scores from k-fold validation, we find that the Xception model does not generalise as well as expected to the unseen data, while the MobileNet and Inception architectures do so well, with a higher accuracy and F1 score achieved by both, due to an improvement in recall score. This could be due to the addition of the extra images made available from dataset 1 by not requiring a portion for validation.

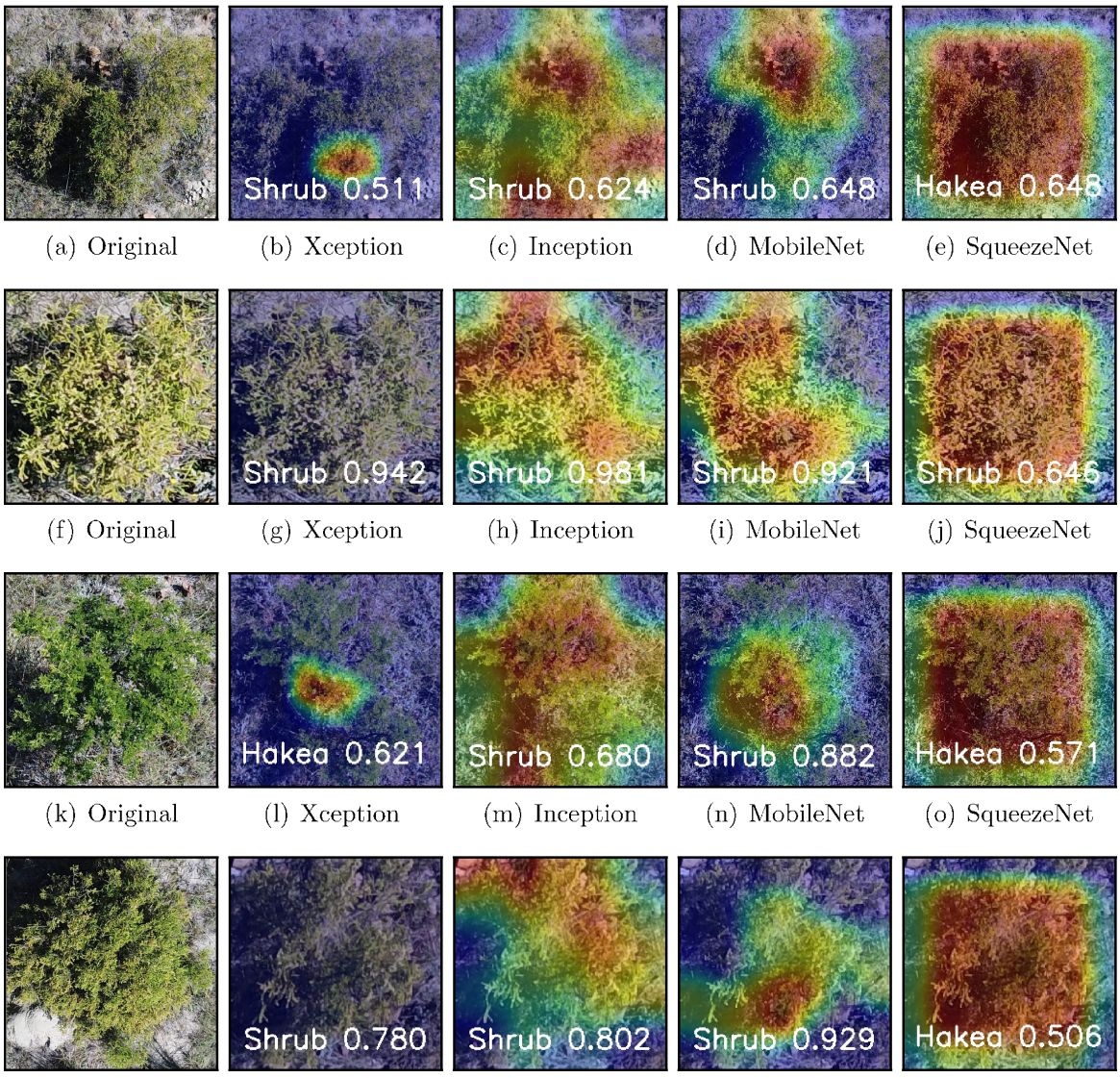
5.2.4 Class activation maps

To further examine the performance of the various models, the class activation maps (also known as heatmaps) of the global pooling layer were examined. Heatmaps were produced using the inference models assessed in the section above for a sample of illustrative images, which are shown in Figures 5.2 and 5.3. These heatmaps show the areas of the image that the respective model activates for that particular image.

The Xception model produces poor heatmaps, as they are very localised in distribution and do not activate over the entire shrub or in some instances do not activate at all. This suggests that although the model achieves good accuracy, it does this using a small set of features extracted from only a portion of the image, rather than the whole shrub. In contrast, the Inception model tends to activate over a large portion of the image, activating well over the objects of interest, although slightly distended surrounding them, which may indicate some use of the area surrounding the shrub in the classification. The MobileNet model also produces successful heatmaps that activate well over most of the objects of interest and are less distended than those produced by the Inception model. The SqueezeNet model produces disappointing heatmaps, as it appears to have learnt to activate over the same region of the image, no matter the position of the object of interest. Therefore, the heatmaps produced indicate that the Inception and MobileNet models are potentially the most reliable, as it is clear that their predictions are produced from activations that extend over the shrub of interest.

5.2.5 Choice of most optimal classifier

Based on the accuracy and F1 scores obtained during cross-validation, the Xception and MobileNet models are the strongest candidates, although it is suggested by the class activation maps that MobileNet may be the most reliable. This is also substantiated by its strong performance on the unseen test set, indicating its ability to generalise well.



(a) Original (b) Xception (c) Inception (d) MobileNet (e) SqueezeNet

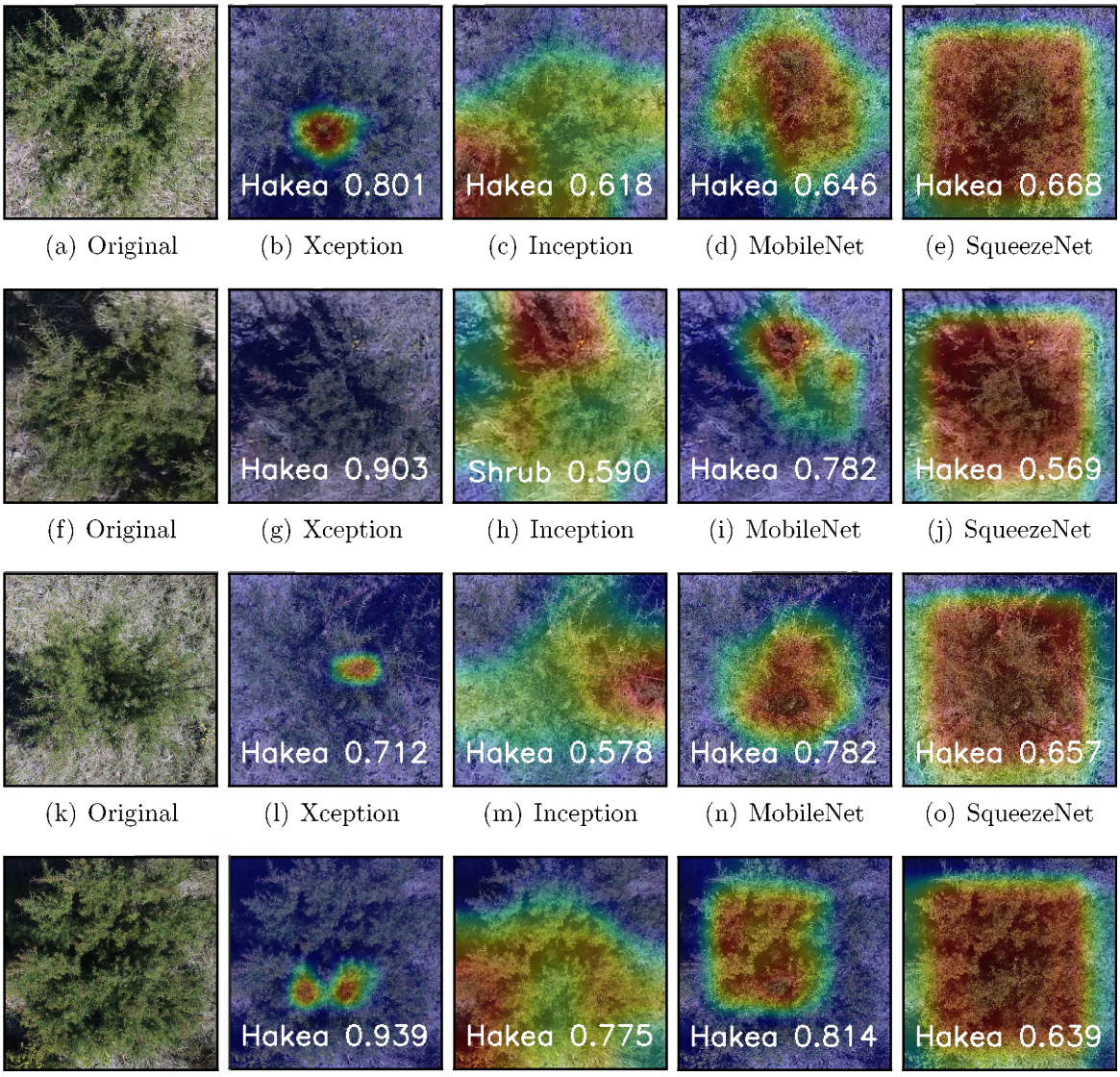
(f) Original (g) Xception (h) Inception (i) MobileNet (j) SqueezeNet

(k) Original (l) Xception (m) Inception (n) MobileNet (o) SqueezeNet

(p) Original (q) Xception (r) Inception (s) MobileNet (t) SqueezeNet

Figure 5.2: Class activation maps for a sample of images from the other class, where correct predictions are those labelled "shrub".

Hence, this model was selected as the final model for use in the research documented in Sections 5.3 and 5.4 and is suitable for use when inference needs to be performed on dataset 2 as an unseen test set.



(a) Original (b) Xception (c) Inception (d) MobileNet (e) SqueezeNet

(f) Original (g) Xception (h) Inception (i) MobileNet (j) SqueezeNet

(k) Original (l) Xception (m) Inception (n) MobileNet (o) SqueezeNet

(p) Original (q) Xception (r) Inception (s) MobileNet (t) SqueezeNet

Figure 5.3: Class activation maps for a sample of images from the target class, where correct predictions are those labelled "*Hakea*".

5.2.6 Summary of findings from preliminary investigation

The performance of the Xception, Inception, MobileNet and SqueezeNet models when fine-tuned using dataset 1 provide baseline accuracies for assessing the task of differentiating between the target (*Hakea*) and other (non-*Hakea*) shrub classes.

From the metrics collected over 10-fold cross-validation, it is observed that the average validation accuracy ranges from 75 ± 0.13 to 0.90 ± 0.07, while the test accuracy ranged between the slightly lower values of 0.72 and 0.98. These accuracies show acceptable

performance for the classification task by all models, although Xception, Inception and MobileNet produce markedly better performance than SqueezeNet.

The heatmaps produced by each model when trained for inference provided insight into the areas of the image used to make the classification. The best heatmaps were produced by MobileNet and the Inception model. The heatmaps produced by the Xception model were meaningless, activating on only small portions of the object of interest or other areas of the image and often not indicating a focus on any particular part of the image, while SqueezeNet also performed poorly, activating over the same portion of the images in each instance.

These calculated baseline accuracies provide a reference for comparison with the model developed in the next section. Of the models, the MobileNet model produced the most correctly focussed heatmaps and this, along with the small difference in accuracy between Xception and MobileNet when using dataset 1 only and between MobileNet and Inception when using dataset 2, indicates that the MobileNet model is the most suited to the classification task. Hence, in this research, a baseline accuracy of 0.87 ± 0.07 is established for the average validation and 0.98 for the test set.

5.3 Pre-processing effects on accuracy

Until the popularisation of neural networks that were able to learn which features to extract, it was necessary to use image processing methods to extract hand-crafted features prior to classification by shallow learning algorithms. An example of this is the HOG features commonly used by SVM classifiers. Although pre-processing is not strictly necessary for neural networks and deep learning, in this section we empirically examine the effects of pre-processing on the accuracy of the fine-tuned MobileNet classifier developed in Section 5.2 to determine whether an improvement in classification accuracy can be obtained.

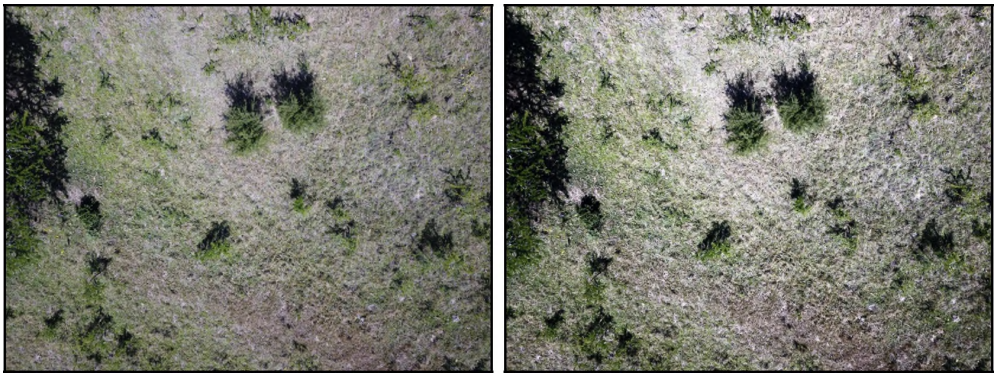
The raw images were obtained under naturally varying environmental and lighting conditions, as explained in Chapter 4. As such, there is distinct variance within each class of the dataset, particularly with reference to the lack of contrast within images when the illumination dimmed due to cloud cover. In an attempt to improve the contrast within the images, and thus classification accuracy, histogram equalisation was assessed.

Prior to extraction of the annotated bounding boxes, histogram equalisation was

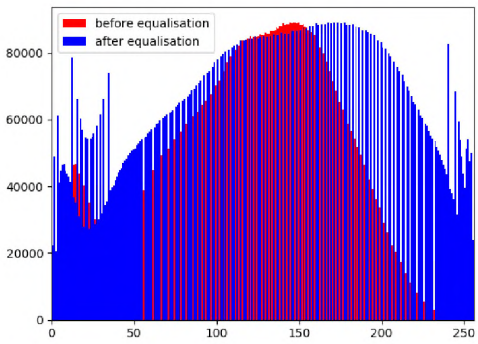
```
def Histogram(frame):
    # equalise the histogram of the Y channel
    image = cv2.cvtColor(frame.copy(), cv2.COLOR_BGR2YUV)
    image[:,:,0] = cv2.equalizeHist(image[:,:,0])
    image = cv2.cvtColor(image, cv2.COLOR_YUV2BGR)
    return image
```

Listing 5.2: Histogram equalisation function.

performed across the entire frame, using the function in Listing 5.2. Equalisation was performed on the Y channel within the YUV colour space, as this channel controls luminance. Hence, by adjusting this channel such that it spans the full range through histogram equalisation, contrast can be improved without affecting the colour component. This is illustrated in Figure 5.4.



(a) Raw image (b) Equalised image



(c) Histogram of Y channel before and after equalisation

Figure 5.4: The change in contrast before and after histogram equalisation in the YUV colour space of the Y channel.

5.3.1 Empirical investigation of effects of pre-processing on classifier performance

The histogrammed cropped sub-images were used to fine-tune the MobileNet architecture with k-fold cross-validation, as in Section 5.2.2. Results of this are presented in Table 5.4, which includes the results for the MobileNet model trained with un-equalised images from Section 5.2.2 for comparison.

Table 5.4: Mean metrics with standard deviations calculated from 10-fold cross-validation of MobileNet model with and without equalisation

Histogram Eq.	Accuracy	Precision	Recall	F1 score
False	0.87 ± 0.07	0.99 ± 0.02	0.75 ± 0.14	0.85 ± 0.10
True	0.82 ± 0.12	0.98 ± 0.05	0.67 ± 0.22	0.78 ± 0.17

A model for inference was then trained on of the histogrammed version of dataset 1, with all images used for training and none for validation. Subsequently, it was applied to dataset 2 to determine how well it generalised to the unseen dataset. Results for MobileNet from Section 5.2.3 are included for comparison in Table 5.5.

Table 5.5: Performance of MobileNet model (with and without equalisation) on the unseen test set, dataset 2

Histogram Eq.	Accuracy	Precision	Recall	F1 score
False	0.98	1.00	0.95	0.97
True	0.97	0.99	0.95	0.97

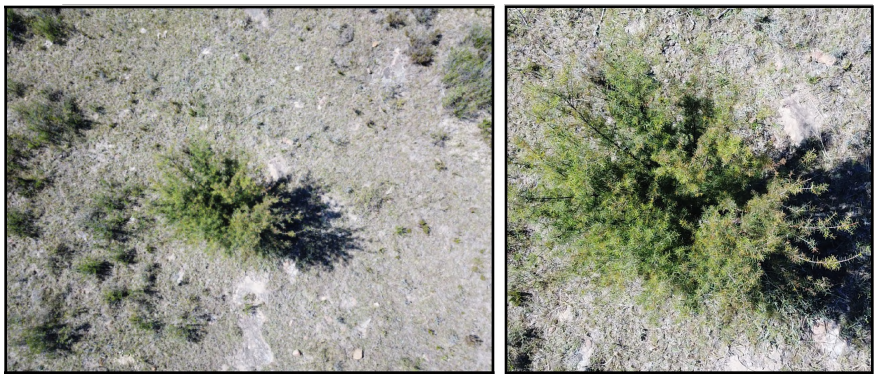
When comparing the metrics obtained through k-fold validation, we find that the performance is poorer on average than when fine-tuning without equalisation. Similarly, the performance of the model for inference is also slightly poorer in accuracy.

5.3.2 Histogram equalisation findings

Histogram equalisation was assessed to determine whether equalising the brightness across the image would improve the classification accuracy. Performance of the model was found to be negatively impacted, and hence we concluded that histogram equalisation does not improve the accuracy of the baseline classifier.

5.4 Candidate proposals

The classifier developed in the previous sections was trained on cropped sub-images of both classes, and as such, similar images are required on which to perform object detection. This requires that candidate windows are proposed from the larger image, as shown in Figure 5.5. This section details the comparison of different methods for segmenting the image to obtain object candidate windows.



(a) Whole frame

(b) Cropped image required for classifier (not to scale)

Figure 5.5: Illustration of the necessity of segmentation: a) shows the whole frame input to the system, while b) shows what the classifier expects, i.e., a bounding box of the image at constant aspect ratio.

There are several approaches to generating candidates to be fed to a classifier, namely, the sliding window method, object proposals and threshold-based methods. Several alternative approaches to the two-stage combination of candidate proposal and classification exist, namely localisation networks and image segmentation networks, which provide all-in-one classification and localisation within a larger frame. Well-known examples of localisation networks, which determine the region of the image in which the object occurs as well as a classification label, are YOLO (Redmon *et al.*, 2016), SSD (Liu *et al.*, 2016), R-CNN (Girshick *et al.*, 2014), Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren *et al.*, 2015). Image segmentation networks assign each pixel in the image to a class, thus performing localisation at a finer grain than simply producing a bounding box containing the object. Since these networks provide an all-in-one solution, they are omitted from this section so that methods for the proposal of candidate windows classification can be examined in greater detail.

5.4.1 Sliding window

Sliding window algorithms are a brute force approach to generating candidate windows for classification. The proposal space is that of the entire image, and all areas of the image are proposed, not only those areas containing bush. The classifiers fine-tuned in Section 5.2 were tuned using only close-cropped images of the target and other shrubs so as to determine the best possible baseline accuracy for distinguishing between the classes. As such, the networks were not trained to distinguish target or other shrubs from other vegetation types which form the background of the image, such as grass, bare ground, rock, fallen or dead bush and shadow. This makes the sliding window approach unsuitable unless another class containing these other vegetation types is added to the fine-tuning process. Hence, the sliding window approach was not considered further.

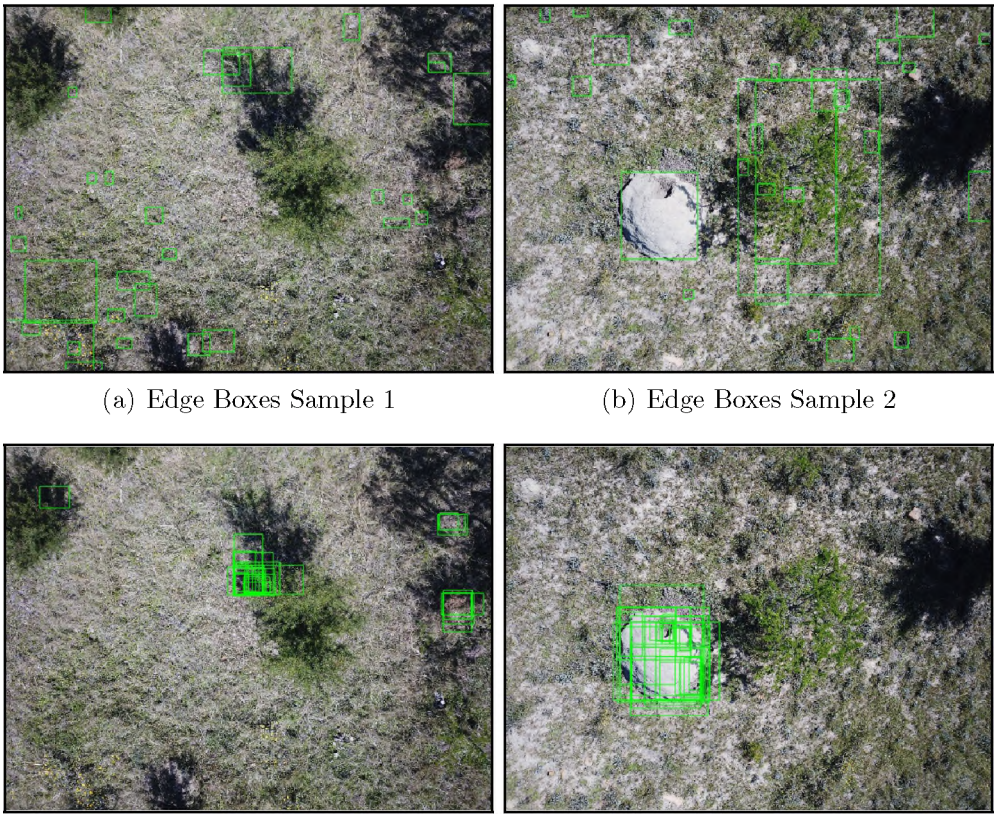
5.4.2 Region proposals

Object proposal algorithms produce bounding box locations in an image which are considered likely to contain an object. This significantly reduces the search space when compared to brute force approaches like the sliding window. At the core of their functionality, object proposal algorithms tend to use either edge detection or superpixels, which are pixels grouped into segments based on compatible properties such as colour and texture. Well-known algorithms based on superpixels are Selective Search (Uijlings *et al.*, 2013) and Randomised Prim's (Manen *et al.*, 2013) while popular edge detection based algorithms are Category Independent Object Proposals (Endres & Hoiem, 2010), Constrained-Parametric Min-Cuts (Carreira & Sminchisescu, 2010), Binarised normed gradients (Cheng *et al.*, 2014) and Edge Boxes (Zitnick & Dollár, 2014). The Objectness algorithm created by Alexe *et al.* (2012) makes use of a combination of edge and superpixel properties and Rahtu *et al.* (2011) build on this algorithm to improve performance. Rantalankila *et al.* (2014) also used a composite algorithm, combining Selective Search and Constrained-Parametric Min-Cuts. For more detail on these algorithms, see the review by Hosang *et al.* (2014).

Of these algorithms, Selective Search is the most commonly used and forms the basis for some of the modern-day localisation networks, such as R-CNN (Girshick *et al.*, 2014). Selective Search builds up object proposals by grouping adjacent superpixels in a hierarchical manner, building larger segments from smaller ones. Edge Boxes is also a highly competitive algorithm, which Zitnick & Dollár (2014) found capable of matching the recall of Selective Search. This algorithm uses the number of enclosed contours as an indication

of the likelihood of an object occurring there.

To determine whether object proposal algorithms are likely to provide the recall necessary when applied to our dataset, one algorithm of each type was applied to the dataset, namely the Selective Search and Edge Boxes algorithms, chosen for their high reported recall. Both of these algorithms are available from **OpenCV**'s extended image processing library, *ximgproc*[2], which was used for testing. Each algorithm was applied to the dataset with two sample images from each presented in Figure 5.6, where only the 30 most probable proposals are displayed on each image. Selective Search was run in 'quality' mode.



(a) Edge Boxes Sample 1 (b) Edge Boxes Sample 2

(c) Selective Search Sample 1 (d) Selective Search Sample 2

Figure 5.6: An example of the results generated through application of object proposal algorithms to two of the images in the dataset.

The Selective Search algorithm was found to primarily propose areas of background vegetation or small sections of bush which were dense in appearance while missing the majority of bushes. As such, a low recall rate was present over the dataset, and the algorithm was found to be unsuitable for proposing candidates on this dataset. The Edge Boxes algorithm was similarly found to be unsuitable, as although the algorithm does pick up some

[2] `https://github.com/opencv/opencv_contrib/tree/master/modules/ximgproc`

of the bushes in the image, it misses the majority and tends to focus rather on objects with a more solid structure and hence clearer edge, such as rocks and bare ground. Hence, we conclude that the nature of the dataset makes the use of object proposal algorithms unsuitable, due to the objects of interest having unclear edges and properties of the target plants lacking sufficiently disparate superpixel properties to that of the background vegetation.

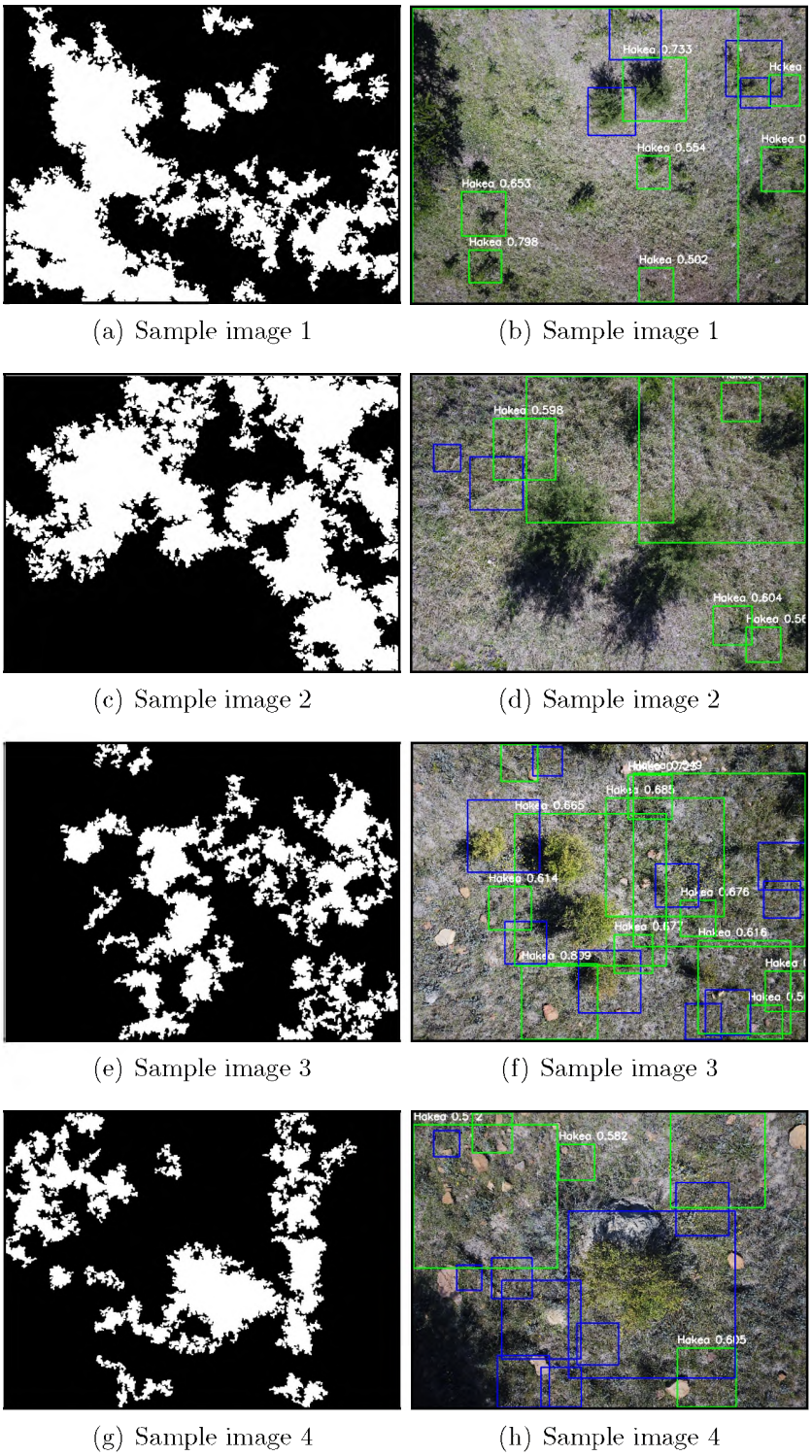
5.4.3 Threshold segmentation

Thresholding was the second method assessed to generate object proposals, particularly HSV thresholding, which splits the image based on its colour range, Otsu thresholding, which uses the image's histogram to segment the image, and vegetation index based thresholding. In each instance, the contours of the thresholded image were found and filtered by area, retaining only those with an area greater or equal to 1000, as this threshold was found experimentally to remove areas too small to be of interest. The binary images were improved upon using the morphological 'open' operation, the contours of the binarised image found and their bounding box coordinates calculated. These bounding boxes were adjusted so as to have an aspect ratio of 1:1, with the smaller of the dimensions being adjusted to match the larger of the dimensions, with equal expansion on either side. If the resulting bounding box exceeded the dimensions of the image by up to half the width or height of the bounding box, then the coordinates of the bounding box were shifted such that the side exceeding the dimensions of the image rested on the edge of the image. Any bounding box exceeding the dimensions of the image by more than half the width or height of the bounding box was discarded. This process is described in Listing 5.3.

The portion of the original image that fell within the bounding box was then extracted for input to the classifier. All inference was done using `mobilenet.h5` from Section 5.2 and all images upon which inference was performed were from dataset 2 and thus unseen to the model.

HSV segmentation

To determine the general distribution of HSV values across the target class, a sample of images from the training set was selected, and the images were cropped to contain only



(a) Sample image 1 (b) Sample image 1

(c) Sample image 2 (d) Sample image 2

(e) Sample image 3 (f) Sample image 3

(g) Sample image 4 (h) Sample image 4

Figure 5.7: Masks (left) and predicted classifications (right) as a result of HSV segmentation.

```
x, y, width, height = cv2.boundingRect(c)
if(width > height):
  d = width-height
  height += d
  y = int(y-d/2)
  half = int(height/2)
  if(y+height - 600 > half or y < -half):
    continue
  if(y+height >= 600):
    y= 599 - height
  if(y < 0):
    y=0
else:
  d = height-width
  width += d
  x = int(x-d/2)
  half = int(width/2)
  if(x+width - 800 > half or x < -half):
    continue
  if(x+width>= 800):
    x = 799 - width
  if(x<0):
    x=0
  if(x < 0 or x + width >= 800 or y < 0 or y >= 600):
    continue
```

Listing 5.3: Aspect ratio adjustment and shift if less than half out of the frame.

the target plants and no surroundings. These 73 resulting images were used for analysis. It was found that the minimum and maximum HSV values were [0,0,0] and [179,255,255] respectively, while the average for each channel was [44.36, 63.62, 133.53], with a standard deviation of [31.33,48.31,70.30] for each channel. These values were used as a starting point to determine the upper and lower thresholds for HSV segmentation. The upper and lower bounds of the H and V were set by respectively adding and subtracting the standard deviation from the mean. This was also tried for the S channel but was found to yield poor results. A lower bound of 30 and upper bound of 230 were experimentally found to give a better result. The final lower and upper bounds were [13,30,63] and [76,230,204] respectively. The resulting mask was filtered to remove small contours as described at the beginning of this section, and a sample of the resulting masks can be seen in Figure 5.7 (a)-(d) with the classifications of the resulting bounding boxes given in Figure 5.7 (e)-(h).

Otsu thresholding

Otsu thresholding is suitable when there are two distinct peaks within the histogram of the image as is the case in Figure 5.8.

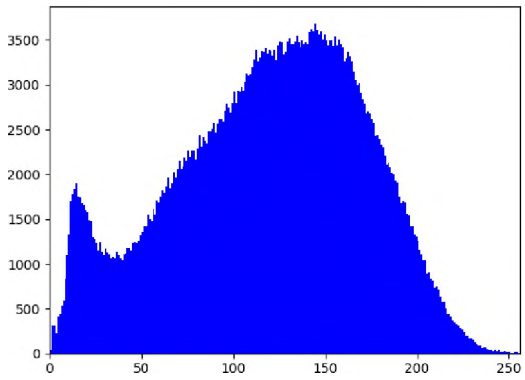


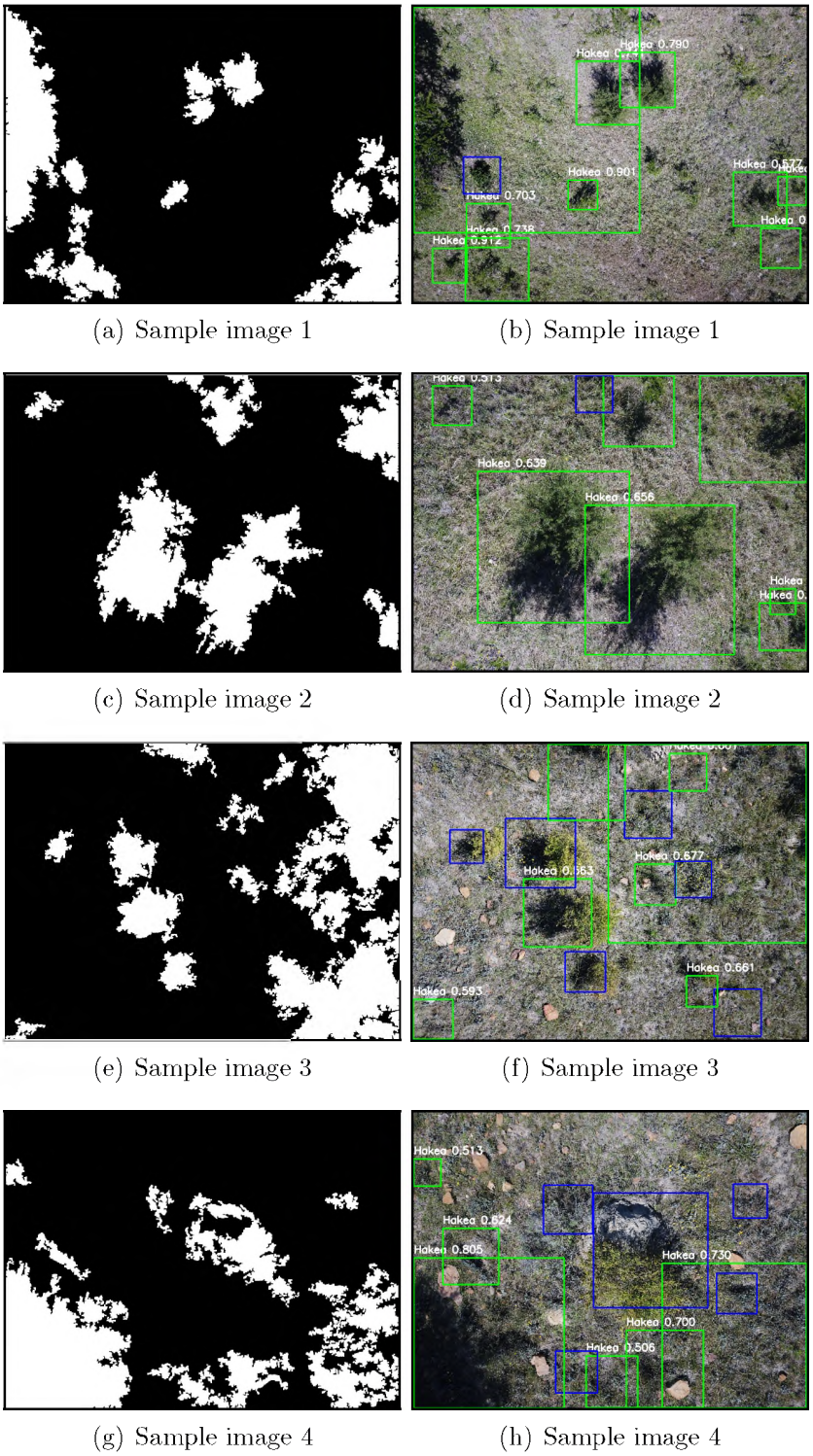
Figure 5.8: Histogram showing two distinct peaks, indicating suitability for Otsu thresholding.

The image was thresholded using Otsu thresholding, with binary thresholding inverted, such that the candidate target shrubs were masked white, as shown in Figure 5.9. Considering the binary image produced, Otsu thresholding picks up the shadows as well as the bushes, which is not ideal. Since the classifier was not trained to distinguish shadows from bushes, this is likely to confuse the classifier. This can also result in large areas of the ground being masked as candidates if there is a variation in lighting across the image.

Vegetation indices

The vegetation indices suitable for use with only visual wavelength light are ExG, ExGR, CIVE, VEG, COM, NGRDI, GCC, WI, VARI and GLI, as discussed in Section 2.3.3. A single frame was used to perform an initial assessment of the effectiveness of these vegetation indices for determining the position of bushes within the image. Each of these indices was calculated for each pixel within the image, transformed to the range $[0 : 255]$, which is the range of pixel values in the image and then plotted using a 'spectral' colour scale with *matplotlib*[3]. The spectral colour scale makes it easier to distinguish visually between the different pixel values. The resulting plots are shown in Figure 5.10.

[3] `https://matplotlib.org/`



(a) Sample image 1 (b) Sample image 1

(c) Sample image 2 (d) Sample image 2

(e) Sample image 3 (f) Sample image 3

(g) Sample image 4 (h) Sample image 4

Figure 5.9: Masks (left) and predicted classifications (right) as a result of Otsu segmentation.

These plots were visually examined to determine which of these vegetation indices were the most successful in separating the bushes from other vegetation types, such as bare soil, grass and rocks. ExG, CIVE, GCC and WI all clearly indicated the position of the target shrub without picking up its shadow as well, whereas the NGRDI, VEG and COM indices did not distinguish the target shrub's shadow from the bush itself. The remaining vegetation indices produced images with too much noise to be useful.

For each of the more successful indices, thresholding was performed to assess the feasibility of generating candidate proposals for the classifier. OpenCV's `inRange` function was used, and the initial values for the range were visually determined from the spectral plots above. These values were then slightly adjusted by trial and error to obtain the best segmentation of the image. The final range values used are given in Listing 5.4.

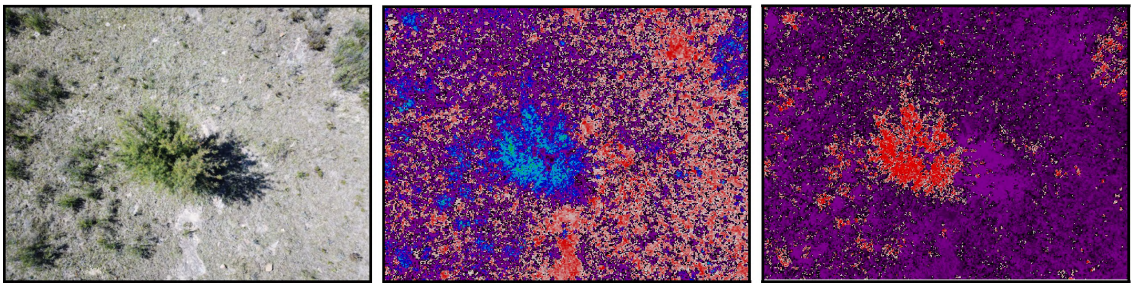
```
1 exg     =  cv2.inRange(exg,30,100)
2 cive    =  cv2.inRange(cive,210,255)
3 gcc      =  cv2.inRange(gcc,90,160)
4 wi      =  cv2.inRange(wi,0,1)
```

Listing 5.4: Threshold values for each promising index.

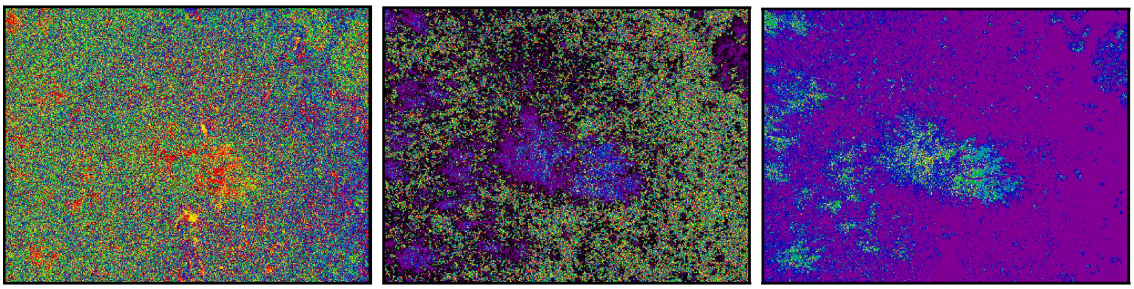
As in the case of both the HSV and Otsu segmentation, the binary images were improved using the morphological 'open' operation, followed by the use of a minimum area criterion of 1000 for contour area and adjustment of the aspect ratio of the bounding boxes to 1:1. The results of the filtered binarisation and the resulting classifications are presented in Figures 5.11 – 5.14.

From visual inspection of the masks of several images, the best performing indices are GCC and ExG, as these indices pick up the majority of the bushes, which are candidates for classification. GCC tends to result in slightly fuller bushes that are less prone to holes than ExG, and although GCC is also inclined to pick up textured ground cover as candidates as well, it detects candidates much more reliably than ExG. The CIVE index is overly conservative and misses a large proportion of the bushes, while WI picks up a large amount of ground cover in addition to bushes. The CIVE index performs better when the bushes are smaller within the field of view, although a number of bushes are still missed altogether. Hence, the GCC index is the best for proposing candidate bushes.

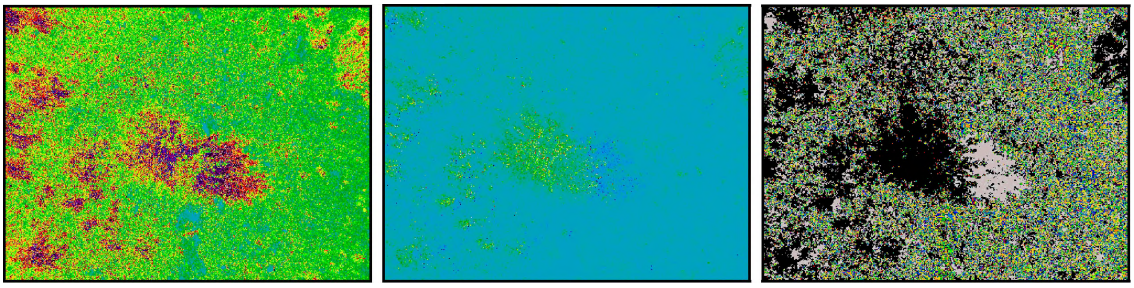
Although GCC performed well at proposing bounding boxes for classification, these bounding boxes were often imperfect when compared to the ideal close-cropped ones used in training. It is also common for individual bushes that are touching to be proposed as



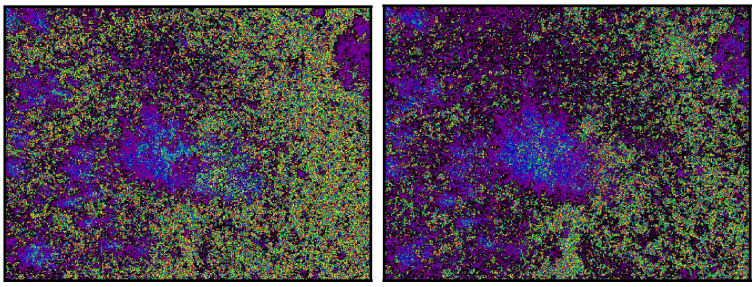
(a) Original frame (b) ExG (c) CIVE



(d) ExGR (e) NGRDI (f) VEG



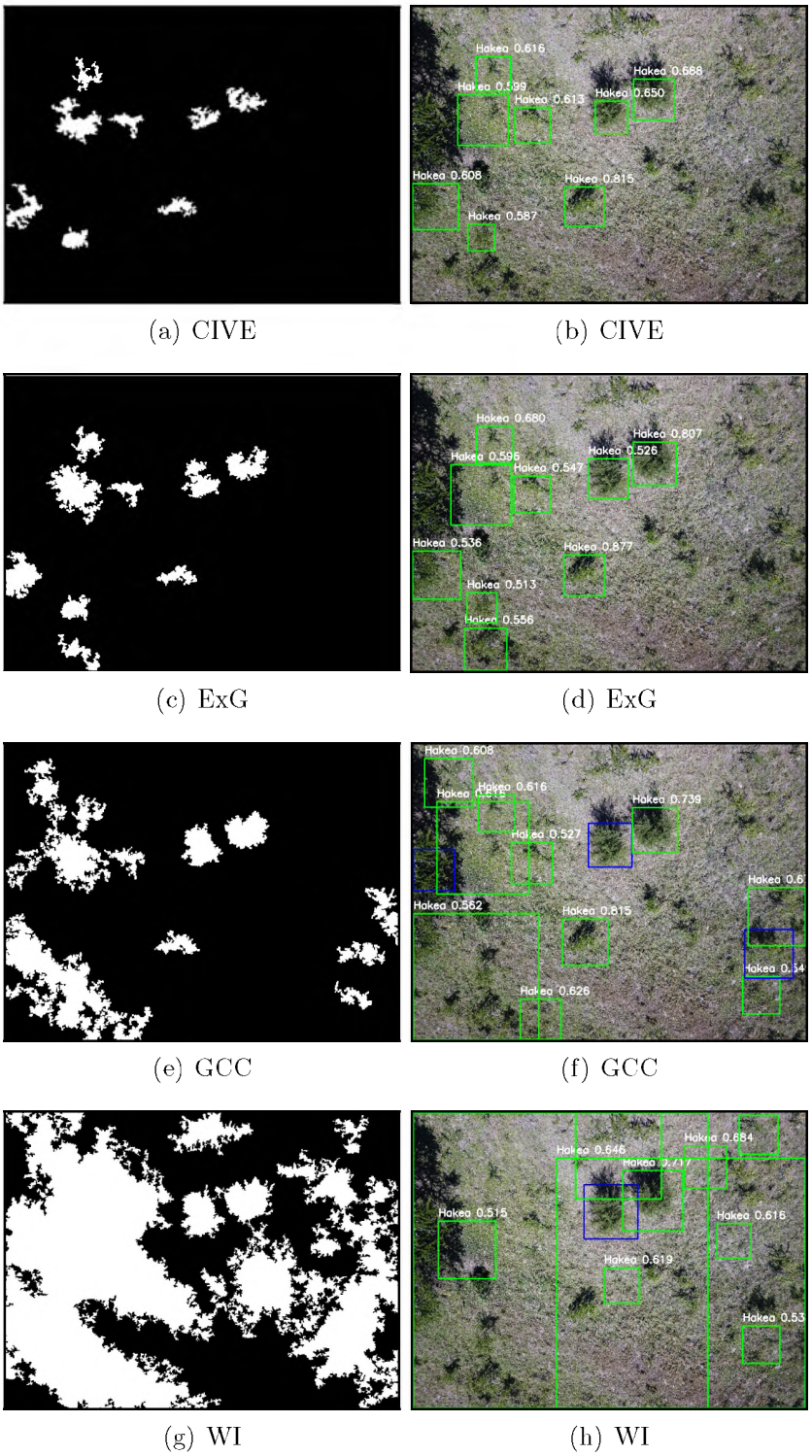
(g) COM (h) GCC (i) WI



(j) VARI (k) GLI

Figure 5.10: Plots resulting from calculation of the various vegetation indices.

a single bounding box, rather than two separate ones. Both this and the imperfect proposals resulted in a high rate of mis-classifications of portions of the image as the target class. To make the use of GCC feasible, the classifier would need to be tuned further using the imperfect bounding boxes.



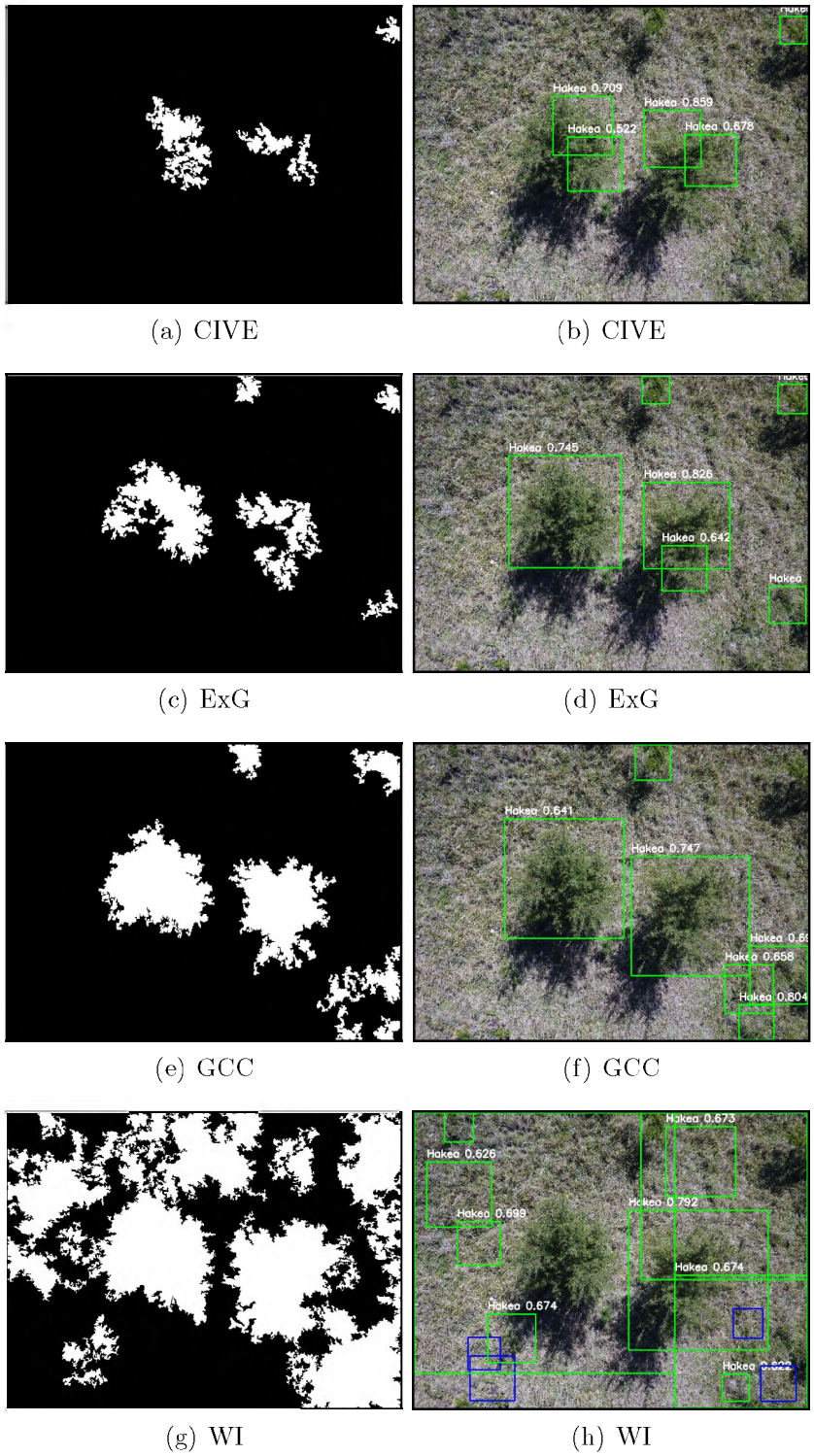
(a) CIVE (b) CIVE

(c) ExG (d) ExG

(e) GCC (f) GCC

(g) WI (h) WI

Figure 5.11: Plots resulting from calculation of the promising vegetation indices (Sample image 1).



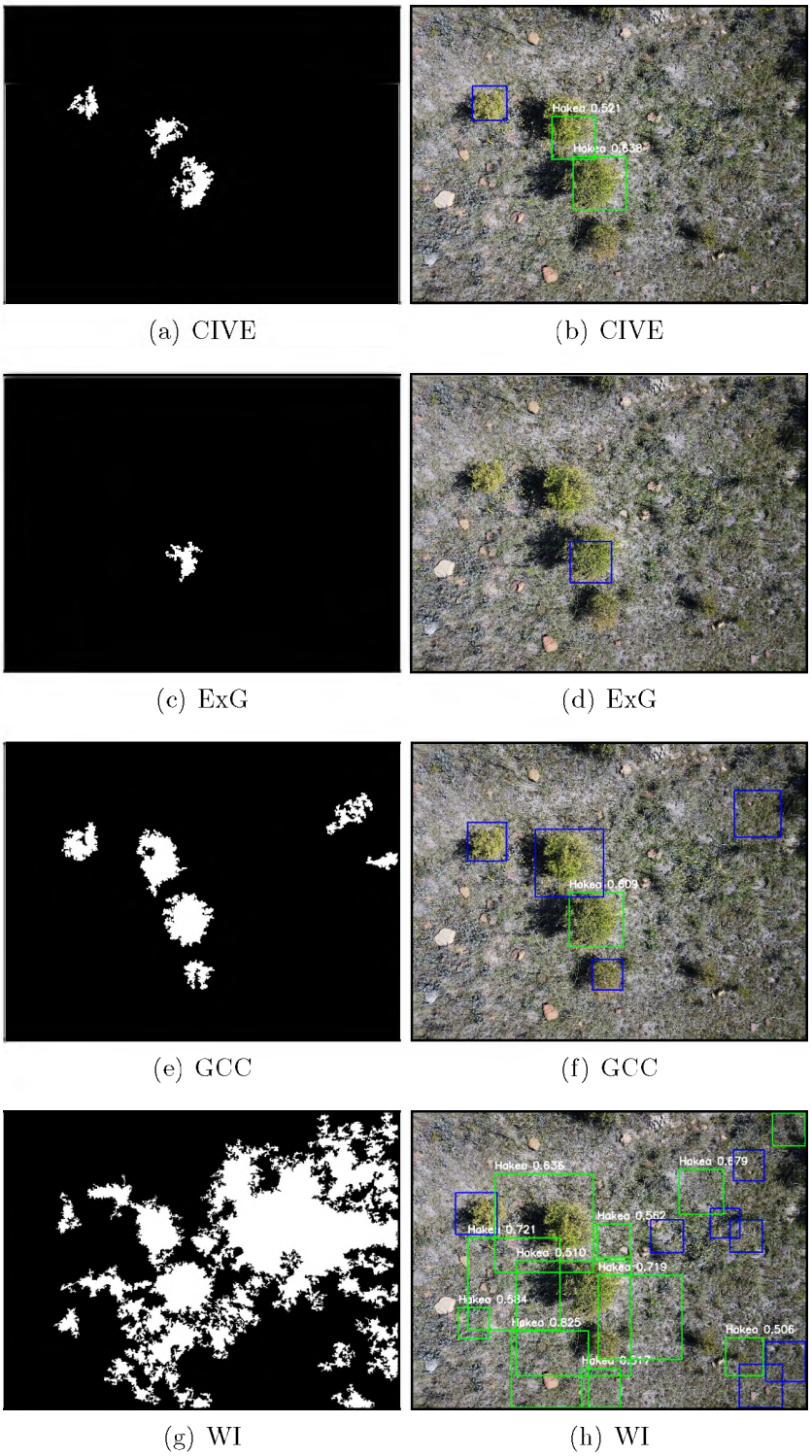
(a) CIVE (b) CIVE

(c) ExG (d) ExG

(e) GCC (f) GCC

(g) WI (h) WI

Figure 5.12: Plots resulting from calculation of the promising vegetation indices (Sample image 2).



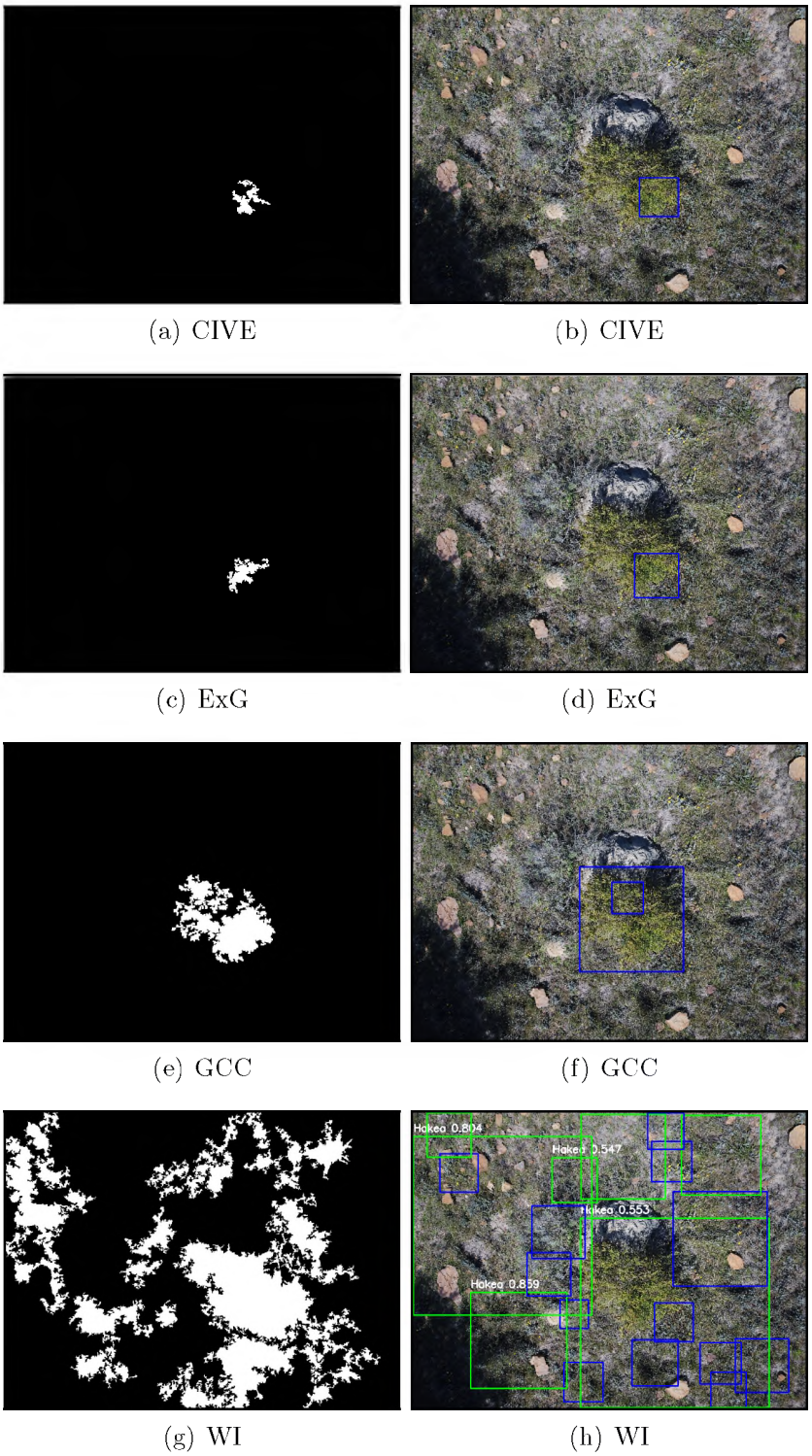
(a) CIVE (b) CIVE

(c) ExG (d) ExG

(e) GCC (f) GCC

(g) WI (h) WI

Figure 5.13: Plots resulting from calculation of the promising vegetation indices (Sample image 3).



(a) CIVE (b) CIVE

(c) ExG (d) ExG

(e) GCC (f) GCC

(g) WI (h) WI

Figure 5.14: Plots resulting from calculation of the promising vegetation indices (Sample image 4).

Comparison of thresholding for segmentation

The HSV segmentation has the benefit that the majority of detections are indeed bushes, with shadows on the ground generally not picked up in the masking process. However, HSV segmentation has the downside that candidates for the target class are missed when the colour value falls outside the range. In contrast, slight variations in colour do not have such a significant effect when segmenting with the Otsu method. Unfortunately, when the lighting across an image changes dramatically, this shifts the histogram and results in incorrect segmentation with the Otsu method. For both methods, there are a number of candidate windows which do not fall on bushes and instead fall on other vegetation types. The classifier was not trained to distinguish shrubs from other vegetation types, such as grass or bare ground, and as such if these thresholding methods were utilised in earnest, it would be necessary to either conduct hard negative mining, to retrain the classifier with a third class containing grass and bare ground, or to create a classifier cascade that eliminates non-bush candidates and as such is not a feasible option for producing reliable candidate proposal windows as it stands.

The GCC vegetation index performed better than the HSV or Otsu methods for segmentation, providing more accurate bounding boxes than either of the other methods, with fewer candidates missed and fewer areas of background vegetation proposed. However, due to the pixel by pixel transforms that are needed to compute the vegetation indices, this method is considerably slower than either HSV or Otsu. Classifications made on the bounding boxes tended towards incorrect classification as the target class, due to imperfect bounding boxes and fine-tuning the classifier further using imperfect bounding boxes would be necessary to make use of the GCC index viable as a candidate proposer. The GCC index does, however, show promise as a candidate proposer.

5.5 U-Net for image segmentation

Image segmentation is the process of assigning a class label to each pixel, as opposed to a single value prediction as in the case of classification. An image segmentation algorithm that has risen to attention is the U-Net. U-Net, originally designed by Ronneberger *et al.* (2015) with biomedical image segmentation in mind, is a modified and extended version of an FCN, which contains no fully connected layers. This network was designed to utilise small datasets for training and to produce precise segmented images in a short run-time.

According to Ronneberger *et al.* (2015), the architecture consists of a contracting part to perform contextualisation, and an expanding part to perform localisation. The contracting part operates in the same manner as a normal CNN, with two 3×3 convolutional layers followed by pooling layers repeated in the architecture, resulting in a shrinking feature map, while the expanding part replaces these pooling layers with up-sampling layers which increase the size of the feature map again. The high-resolution features from the contracting path are combined with the up-sampled output of the expanding path to perform localisation, and a final 1×1 convolutional layer maps the resulting feature map to the classes. This system relies on a high level of data augmentation, which allows the network to develop robustness against deformations in the images.

Bouchareb (2018) used the U-Net architecture to perform candidate segmentation of bushes from background land cover in aerial imagery. Bourchareb's research differs from that presented in this section, as in this section we wish to determine whether it is possible to train the network to distinguish between our target shrub and other vegetation, rather than between any shrub and background vegetation.

5.5.1 Mask creation

According to Ronneberger *et al.* (2015), the U-Net architecture requires images of the size 512×512, with an aspect ratio of 1:1. To achieve this aspect ratio, the input images of size 3000×4000 were cropped so as to prevent distortions, which would be the result of resizing a different aspect ratio. Five hundred pixels were cropped from each side of the longer side to produce an image of 3000×3000 pixels. These cropped images were then semantically annotated using GIMP to produce ground truth segmentation masks, with pixels containing the target class coloured white and the remainder of the image coloured black.

5.5.2 Binary cross-entropy loss as a baseline

A core aspect of training a deep learning model is the loss function, which computes a measure of error between the predicted output and the annotation mask. A simple and commonly used loss function is the cross-entropy (CE) loss, which is defined in Equation 5.1, where N is the number of samples and C is the number of classes, calculated over the predicted volume $p_{nc} \epsilon P$ and ground truth volume $g_{nc} \epsilon G$.

$$L_{CE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} g_{nc} log(p_{nc}) \tag{5.1}$$

In this chapter, we address a two-class, or binary, segmentation problem, where pixels belonging to the background class have value 0 and those belonging to the target class have value 1. The loss, in this case, is referred to as binary cross-entropy (BCE) loss.

To train the model, the Keras implementation of BCE was used. This function takes N to be all the samples within a batch; hence BCE loss is averaged across all samples in a batch, rather than obtaining a single value for each image within the batch and averaging across these.

5.5.3 U-Net configuration

The parameters and initialisations chosen were those given in the original U-Net paper (Ronneberger *et al.*, 2015), namely stochastic gradient descent with momentum 0.99 and BCE loss, although the original paper uses a modified BCE loss. Weights were initialised using the `he_normal` option in Keras, which draws weights from a Gaussian distribution with a standard deviation of $\sqrt{\frac{2}{N}}$, where N is the number of inputs to a neuron. Additionally, 16 filters and a 0.05 drop-out were used.

One of the strengths of the U-Net is its low requirement for the number of training samples, instead of relying on heavy data augmentation. To achieve sufficient data augmentation, an *ImageDataGenerator* was used. The deformations made through data augmentation allow the classifier to learn to some extent invariance to flips, shifts, rotations and scale, which, as stated by Li *et al.* (2018) are particularly necessary for remotely sensed data.

5.5.4 Training scheme

Ten-fold validation was performed to obtain a reliable measure of the model's performance, which was computed through a series of metrics. The model outputs a probability heatmap, which was thresholded at 0.5 to separate the class predictions for each pixel. The morphological 'open' operation was then performed on the image to remove any noise in the output prediction, producing the final prediction from which the metrics were calculated. Thereafter, a model was trained for inference, using the whole of dataset 1 as a

training set, with dataset 2 used as an unseen test set.

5.5.5 Metrics

The metrics defined in Section 2.4.2, namely precision, recall, F1 score and accuracy, are once against used to gauge performance. Contrary to the classification task discussed in Section 5.2 which uses only one-dimensional annotation label vectors, in image segmentation problems, the annotation masks are two-dimensional. This means that the metrics can be calculated either across the entire batch, irrespective of the distribution of positive and negative predictions across the batch, or per image and then averaged across all images in the batch. We compute both sets of metrics, as they each provide a different insight into the model behaviour, with the first referred to as 'batchwise' metrics and the latter as 'average' metrics.

5.5.6 Performance evaluation

As shown in Table 5.6, very similar results were obtained for both batchwise and average metrics. The global accuracy achieved is a notably high score, indicating that a vast majority of pixels were predicted correctly. However, all other metrics achieved markedly lower values than this. The precision value indicates that of all predicted positives, 87% on average of those predicted positives were true positives, while the remainder were false positives, that is, other shrubs predicted to belong to the target shrub class. The recall shows that on average, only 56% of all ground truth positive pixels were detected correctly.

Table 5.6: K-fold metrics – using BCE loss applied to dataset 1

	Precision	Recall	F1	Dice	IOU	Accuracy
Batchwise	0.87±0.05	0.56±0.22	0.72±0.09	0.65±0.23	0.51±0.20	0.96±0.01
Average	0.70±0.14	0.56±0.21	0.72±0.06	0.49±0.21	0.41±0.18	

These observations indicate that although the accuracy of the model in terms of the sheer number of pixels predicted correctly is good, there is still a large portion of ground truth positive pixels that are not detected as positive (44%). A reasonable score is achieved for F1, due to the very high precision values which compensate for the moderate recall values. The moderate performance by recall is possibly due to the imbalance of the two classes, as it is possible to minimise loss by favouring the prediction towards the larger, background class.

5.5.7 Inference on unseen test set

To produce a single model for inference on the unseen test set, a single network was trained with BCE loss using the whole of dataset 1, over 150 epochs. The resulting model was then used to predict over dataset 2, the results of which are seen in Table 5.7.

Table 5.7: Inference model metrics – trained using BCE loss on dataset 1 with inference performed on dataset 2 as an unseen test

	Precision	Recall	F1	Dice	IOU	Accuracy
Batchwise	0.93	0.28	0.43	0.43	0.27	0.93
Average	0.80	0.29	0.44	0.33	0.25	

A sharp decrease in performance is observed when the model was tested on the unseen dataset 2. This indicates that there are features that the model has learnt that are part of dataset 1 but are different or not present in dataset 2. This is entirely possible as the two datasets were collected on different days and as such, under different environmental conditions. As the precision is very high, with correspondingly low performance in recall, it is possible that the model has learnt a very small set of features and that when applied to the unseen dataset these features do not adequately match some of those within dataset 2, such as brightness. Nonetheless, when considering Figure 5.15 which is a representative sample of the test set, with the mask, raw and binarised predictions, we see that in some images the target plants are predicted well (columns 1 and 4) and that plants that appear similar to the target shrub are not classified as false positives (column 5). This indicates favourably that the U-Net model appears to have the capacity to learn to distinguish between these plants.

It is seen that each target plant contains at least some area within it that is classified as belonging to the target class, although particularly in column 2 this was not detected particularly well, missing the vast majority of pixels belonging to the target class. One small false positive was detected in column 4, but on the whole, the model's qualitative performance at object level agrees with its pixel-level performance, showing very few positive predictions, although those that were made were almost all correct.

5.5.8 Discussion on the performance of U-Net

The U-Net model was trained with stochastic gradient descent with a high momentum and BCE loss to obtain a baseline model. K-fold validation was used to determine an estimate of its performance on dataset 1, and a model for inference was subsequently

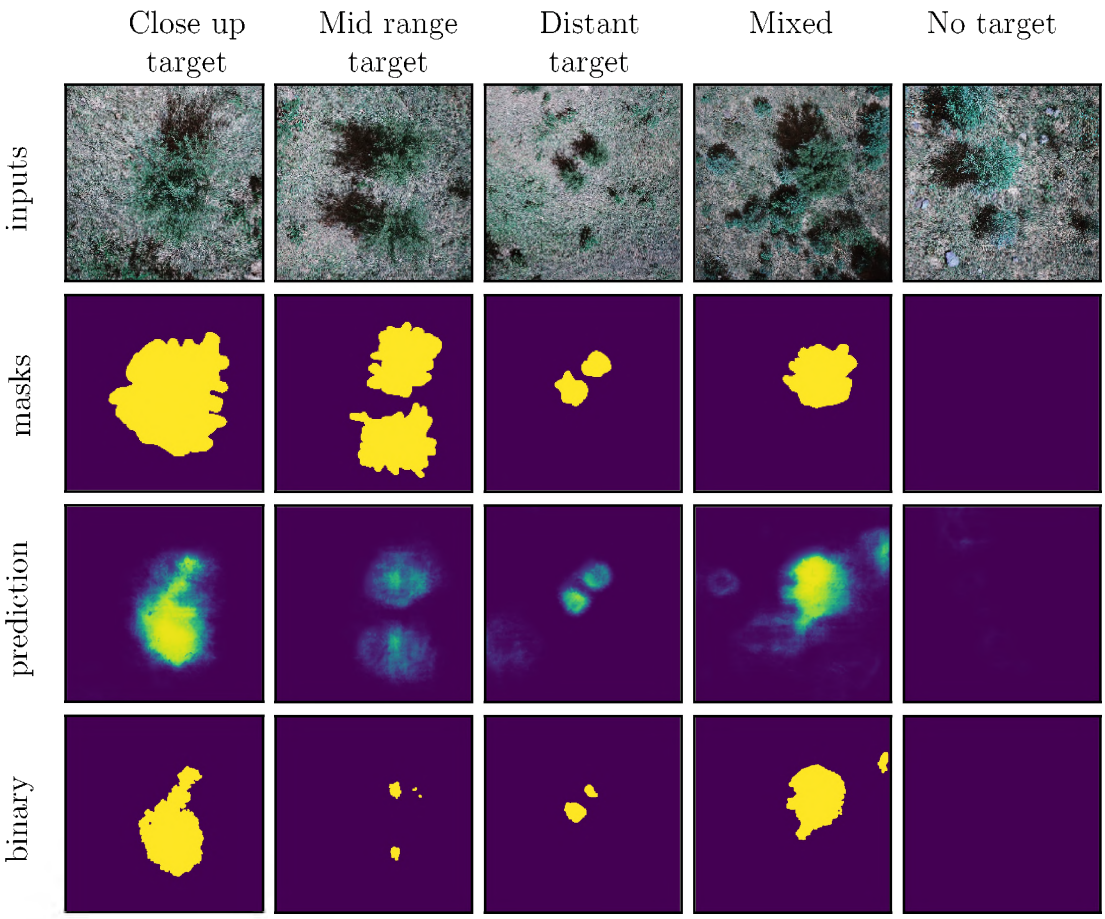


Figure 5.15: A selection of input images (row 1) for different cases of image composition, along with their corresponding binary masks (row 2), their raw predictions (row 3) and the thresholded final predictions (row 4) for the top-performing model.

trained using the whole of dataset 1 and assessed on the unseen test set dataset 2. The model performed well on some images, but poorly on others, with good precision but often poor recall, missing many of the target pixels. This indicates that the U-Net architecture has the potential to detect the target shrub in a natural scene, although further training with perhaps different loss functions is necessary to improve the recall of the model.

5.6 Inference model for field testing

A model was trained for inference in the field using both datasets. The weights for the model trained on dataset 1, used for inference in the previous section were used as a starting point, and the model was trained for a further 150 epochs using dataset 2. This model, having been trained on all available data, is suitable for integration into an application for preliminary field testing.

5.7 Summary

This chapter set out to determine whether a model could be trained to reliably detect the target shrub, *Hakea*, within a frame. This was approached through the questions;

- What baseline accuracy is achievable through fine-tuning existing models?
- Can pre-processing improve upon the baseline accuracy?
- Are candidate segmentation methods able to provide accurate object proposals which could be fed to a classifier and thus form a workable system?
- Is image segmentation a better approach to this problem, classifying pixel-by-pixel into classes as opposed to a proposal and classifier two-stage approach?

A selection of existing models, Xception, Inception, MobileNet and SqueezeNet, were fine-tuned using dataset 1 and found to provide a range of validation accuracies from $78.76 \pm 7.96\%$ to $84.90 \pm 9.16\%$. Visualisation of the class activation maps showed that MobileNet activated most reliably on the target plant. The successful heatmaps and small difference in accuracy compared to the Xception and Inception models resulted in the MobileNet model being chosen to provide a baseline accuracy of 89.78 ± 6.72 on the validation set, a portion of dataset 1, and $92.53 \pm 7.04\%$ on the test set, which was the entirety of dataset 2.

Histogram equalisation was then assessed as a pre-processing function to address variation in illumination due to varying environmental conditions. It was found that raw RGB achieved marginally better average validation accuracy and hence, that there was no benefit to using histogram equalisation as a pre-processing function.

Region proposal algorithms and thresholding techniques were trialled as candidate proposal approaches. The region proposal algorithms Edge Boxes and Selective Search were found to focus on solid objects like rocks or areas of dense background vegetation, missing the majority of the target plant, which made them unsuitable. The threshold segmentation techniques examined, namely HSV thresholding, Otsu thresholding and vegetation index based segmentation proved more promising, although HSV and Otsu segmentation had vulnerabilities to variation in hue, the effect of variation in illumination of the histogram of the greyscaled image and the similarity in features of the background vegetation. Vegetation index thresholding was the most promising thresholding technique but had the downside of being very slow to compute. Classifications on the bounding boxes produced by these methods were often incorrect as the model used for inference was fine-tuned on perfect bounding boxes. A workable proposer-classifier combination could possibly be developed using the GCC vegetation index if the classifier was re-tuned on a selection of imperfect bounding boxes.

The U-Net architecture was used to determine if direct segmentation into classes was a better approach than a separate proposal and classification system. A trained model predicted 0.96% of pixels correctly in the validation set, a portion of dataset 1, and 0.95% in the unseen test set, the entirety of dataset 2. Upon visual examination of the unseen test set, 69% of the target shrubs had significant activations within the masked area, indicating that segmentation has the capacity to detect *Hakea* shrubs in a natural setting, although further analysis would be necessary to eliminate false positives.

In conclusion, a workable *Hakea* detector can be produced either by using the U-Net segmentation architecture to produce pixel-by-pixel predictions of the target versus other classes, although further analysis may be required to eliminate false positive detections. Another workable approach could possibly be produced by combining a GCC vegetation index based candidate proposer with a MobileNet model fine-tuned on imperfect bounding boxes, although this has not been implemented at this stage.

Chapter 6

Integration of model into drone system and preliminary field testing

TensorFlow models are frequently integrated with mobile applications, as is evident by the specific development of TensorFlow Lite for this purpose. There is a growing number of tutorials and implementations on Github for the integration of TensorFlow models with DJI drone systems, such as the integration of Microsoft's Cognitive Service for facial recognition in Android[1] and integration of Tiny YOLO with a Phantom 4 on iOS[2]. However, there are only a few published accounts of integration of a TensorFlow model into a DJI drone system for inference in flight.

The first of these was the integration of a CNN for graffiti detection with a DJI Matrice 100 by Nahar *et al.* (2017). In addition to the DJI Mobile SDK, the DJI Matrice offers access to the DJI Onboard SDK. The Onboard SDK allows flight control and was connected to a Raspberry Pi 3 which forwarded the video livestream to a server on which the TensorFlow model was housed. This system was capable of both detecting the graffiti and estimating its area. Once a detection was made, further commands were relayed to the drone to initiate a spraying routine to cover the graffiti with paint.

In another article, Anar *et al.* (2018) reported that the classification and detection parts of the TensorFlow Android Camera Demo tutorial[3] were integrated successfully with a DJI

[1] `https://github.com/Li-Yanzhi/DJI-CognitiveService`
[2] `https://github.com/khurram18/DJI-phantom-4-pro-yolo`
[3] `https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android`

Phantom 3. The OpenCV image processing library was used to pre-process the images captured by the drone, through noise reduction and maximally stable extremal regions blob detection to detect targets for classification. Targets included objects in classes such as tractor, lake and pier.

This chapter aims to document the development of an application that integrates the segmentation model introduced in the previous chapter with an application capable of performing inference in the field as well as core drone control functionality. Additionally, this chapter presents the preliminary results for in-field inference performed by the integrated model as an initial assessment of the system's suitability for its task of detecting the target plant in the field. This gives insight into the model's strengths and weaknesses and indicates where there is room for further refinement to produce a more robust system.

6.1 Mobile software development kit

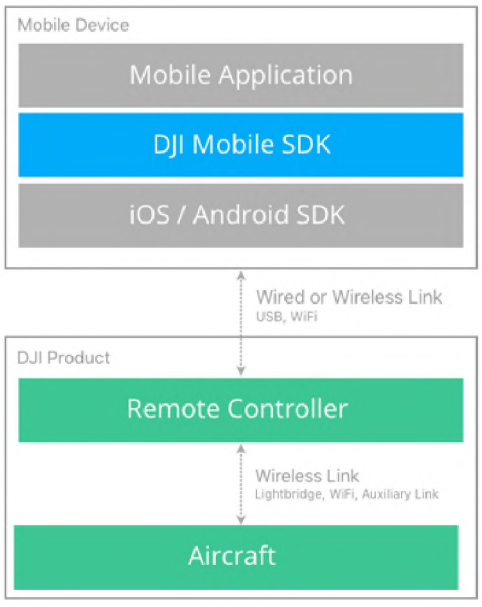
DJI GO is a commercially developed mobile application that allows the user to interact with the drone while it is in flight (DJI, 2019). The application allows the user to view the real-time video feed from the drone, control the camera, perform functions such as take-off and landing and navigate to different flight modes. The video livestream also reports telemetry data, such as aircraft height, distance and battery life.

DJI provides a mobile software development kit (SDK) for Android and iOS to allow developers to make use of user interface (UI) elements for the core functionalities available in DJI GO. This allows the developer to focus on the high-level functionality of the application, building on a mini version of the DJI GO app, rather than focusing on low-level control. The SDK provides access to flight control, telemetry and sensor data, access to the onboard camera and gimbal control, as well as the live video stream and additionally control of pre-defined mission functions.

The SDK has two complementary APIs. The first is the Android SDK, containing manager, base, product, component, mission and miscellaneous classes. A particularly important class within the Android SDK is the *DJISDKManager* class, which is essential for the registration of the SDK as well as connection and access to the drone. Another is the *Aircraft* class, which is accessed from within *DJISDKManager*, and allows access to all components of the drone, such as the camera and flight controller.

The second API is the Android user experience (UX) SDK, which provides access to widget classes. Among these classes is the first person view (FPV) widget, which allows the user to view the video feed from the camera. Widgets for take-off, landing, return to home, battery and altitude monitoring are some samples of those available. It is through these widgets that the user interacts with the drone, in addition to the remote control.

Communication between the Mobile SDK and the DJI product occurs via wireless communication. Lightbridge, a wireless link developed by DJI and which has significantly longer range than regular WiFi, links the aircraft and the remote controller to allow communication in the 2.4 GHz band. This connection allows the transmission of the video feed and application information to the device running the application, while aircraft control information is transmitted through a lower bandwidth, though more robust, auxiliary link. The remote controller is attached to the mobile device by a USB wired connection. The Android SDK obtained the video feed and sensor information and through the Mobile SDK, made this information available as part of the application where they can be displayed on the FPV, and whatever functionality is required by the application can be applied. This is described visually in Figure 6.1. The video feed to the mobile device is H.264 coded and has a resolution of 720p.



(a)

Figure 6.1: Interactions between the drone and the Mobile SDK[4].

[4]Figure reproduced from – `https://developer.dji.com/mobile-sdk/documentation/introduction/mobile_sdk_introduction.html`

6.2 Freezing model for inference

Each Tensorflow model consists of a *GraphDef* object and a set of weights. During training, it is common to save only the weights in either `.h5` or `.ckpt` files; however, when performing inference, it is necessary to combine these two components into a single model file. A process called 'freezing' takes a `.h5` file containing both the weights and the *GraphDef* and replaces all the variable ops with constant values, as well as removing any nodes not used for forward inference. The resulting model is saved as a protobuf file, `.pb`, and can be used for inference. This file is made accessible to the Android application by placing it in the 'assets' folder of the application.

6.3 Libraries

The same functionality available for pre-processing imagery prior to segmentation needed to be available both during the training stage, when Python was used and in the inference stage when Java for Android was used. Both the image processing library used, OpenCV, and the machine learning platform, TensorFlow, met these requirements and are introduced in this section. By using the same library in both environments, we can ensure that variations in standards do not occur.

6.3.1 OpenCV

The Open Source Computer Vision Library (OpenCV) provides support for building real-time machine vision and machine learning applications (OpenCV team, 2019). A vast library of computationally efficient algorithms provides functionality for these applications across C++, Python, Java and MATLAB platforms. This library allowed the necessary functionality for transformations of the images in both the training and inference stages.

In the training stage, this library was invaluable and allowed not only for standard operations such as resizing but also for loading and saving of images from file, brightness transformations and associated colour space conversions and morphological operations on output images to remove noise. In the inference stage, however, the library was used for resizing and extracting a window for segmentation from the larger FPV.

6.3.2 TensorFlow

TensorFlow is a powerful open source machine learning platform that assists in the development and training of machine and deep learning models (Abadi *et al.*, 2016). Various levels of abstraction are available for the development of models, through high-level APIs like Keras. Models can be trained on different processing architectures, namely CPUs, GPUs and Tensor Processing Units.

TensorFlow is made available on Android through the TensorFlow Mobile's *TensorFlowInferenceInterface* class, which is a wrapper for the TensorFlow API providing a small API surface which loads the model and allows easy use thereof through the functions `feed`, `run` and `fetch`.

Recently an alternative to TensorFlow Mobile has become available, TensorFlow Lite, which totes low latency and smaller and faster models as its main advantages over TensorFlow Mobile. However, at the time of initial application development, TensorFlow Mobile was still the most documented option, with TensorFlow Lite still very new on the scene. Future work could incorporate migration to TensorFlow Lite.

6.4 Mobile application architecture

Along with the documentation of the Mobile SDK, several tutorials are available which help developers implement common functionalities, with sample code provided for each tutorial. The application documented in this section was extended from the UXSDKDemo tutorial code[5], which shows how to set up a mini DJI GO application using the UI elements provided in the SDK.

An overview of class relationships in the developed application is presented in Figure 6.2, which was created using the SketchIt![6] plugin in Android Studio to generate a unified modelling language description and then visualised on planttext.com. Each of the important classes and their functions are then introduced in greater detail.

[5] `https://github.com/DJI-Mobile-SDK-Tutorials/Android-UXSDKDemo`

[6] `https://plugins.jetbrains.com/plugin/10387-sketch-it-` or `https://bitbucket.org/pmesmeur/sketch.it`

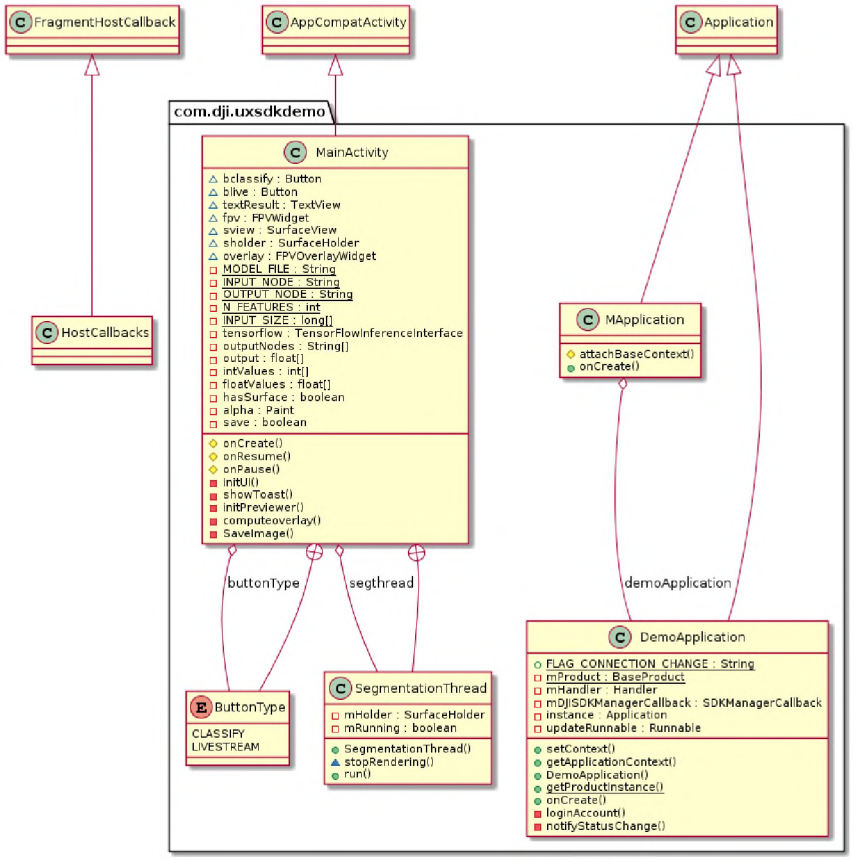


Figure 6.2: Class diagram for extended Android application.

6.4.1 MApplication

This class was supplied as part of the tutorial and exists to load the SDK classes and to invoke the `onCreate()` method of the *DemoApplication* class, discussed below.

6.4.2 DemoApplication

Also supplied as part of the tutorial, the *DemoApplication* class starts the SDK services. It also registers the application using an API key provided by the developer, which was created using their DJI Developer profile and is supplied as meta-data in the Android manifest file. It is also responsible for listening to the connection status of the DJI product, as well as listening for component changes. These listeners are vital to ensure that all connection changes are flagged.

6.4.3 MainActivity

The functionality of the application is provided by the *MainActivity* class and provides the user with the window in which the UI elements that the user can see and interact with are drawn. The FPV widget and other core elements, such as take-off and land, form the base functionality of the application and are provided through the content view. In addition to this, two buttons were added to allow the user to toggle between the default livestream and implemented classify modes.

In livestream mode, the standard mini DJI GO functionality is present. Once the classify button is clicked, a SurfaceView element is overlaid in the bottom right corner of the FPV and a thread to perform segmentation and visualise the results in the SurfaceView is started. The thread continues to sample the FPV at the end of every visualisation until it is interrupted by the selection of the livestream button, which hides the SurfaceView element and stops the thread.

6.4.4 SegmentationThread

SegmentationThread is a class that inherits from the *Thread* class and extends it with the method `computeoverlay`. This method takes a sample from the livestream present in the FPV, pre-processes the resulting bitmap and feeds its pixel values to the Tensorflow inference interface. The output of this inference is then thresholded and used to set the pixel values of a bitmap to either yellow or purple, depending on the resulting class, where yellow indicates a prediction of the target class. The original bitmap used for inference is then resized, and the class bitmap is overlaid over it using a canvas, which is then posted to the SurfaceView element. The overlaid bitmap is also saved to the internal storage of the tablet, for analysis of results. This thread is vital for interacting with the model and was implemented as an extension to the tutorial code.

6.5 Application workflow

The main UI thread continuously updates the FPV with the video livestream, irrelevant of whether in livestream or classify mode. This is essential as the user must be able to see surroundings in both modes so as to safely navigate the drone. Upon entering classify mode, another thread is started to control the classification process, so as not to block the

main thread. A SurfaceView element is overlaid in the bottom right corner of the FPV to provide user feedback on the classification results. In classify mode, the FPV is sampled to obtain a bitmap of size 2048 × 1488 pixels, which is cropped to a constant aspect ratio of 1:1 and then resized to 512 × 512 pixels, as expected by the classifier. The pixel values are obtained, divided by 255 and saved in a 1D float array in pixel component order B, G, R. The resulting float array is then input into the Tensorflow inference interface, which places the inference result in another 1D array. This output array is then thresholded into classes at 0.5 and used to set the values in a bitmap to produce the segmentation mask. The result is saved to the internal storage of the device and also posted to the SurfaceView element. This process is visually depicted in Figure 6.3.

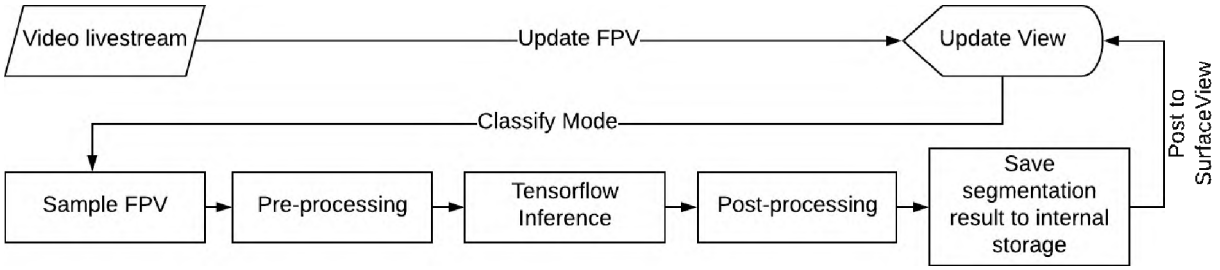


Figure 6.3: Application workflow.

6.6 Application performance

This section details a series of experiments to obtain preliminary results for in-field inference using the application documented in the previous sections. We examine the effects of brightness and altitude on the model's predictions, as well as how brightness effects false positive detection. Additionally, we obtain a measure of the execution time for pre-processing, segmentation and post-processing.

Each of the segmentation outputs included in the results was displayed on the tablet screen while the drone was in the air in the field. Both the input image to the model and the segmentation result were also saved to internal storage for inclusion in this chapter. To interpret the segmentation output, yellow indicates predicted membership of the target class while purple indicates a prediction of the negative class. As a visual aid to the reader, the ground truth instances of the target class were outlined in white for inclusion in this chapter.

6.6.1 Preliminary in-field inference results

Two flights were made, and the segmentation results were saved to internal storage by the application. The tests were conducted in an area from which training data was not collected. This ensures that any possibility of learning individual features of target plants is excluded, although this would be extremely unlikely due to wind deformation of the target plant's branches, growth or damage since capture of training data and other environmental variations. Additionally, by flying in a region not used as training data, we ensure that the model is capable of distinguishing the target plant when the background vegetation varies. The training images were captured in a primarily grassy area, while the area in which the experiments in this section were done was a combination of rocky and grassy.

Variation in image brightness

To determine the extent to which the brightness of the image affects the classification result, a series of tests were performed in which the drone maintained constant position and altitude, while the exposure of the image was manually varied to change the brightness of the image. The results are shown in Figure 6.4.

From inspection of the images, we see that when the image is bright, the model misses the majority of pixels of the target class, and as the brightness decreases the segmentation performance improves in terms of true positive prediction, but with a corresponding increase in false positive predictions. When considering all but the first and last row of images in Figure 6.4, which are extreme cases, we see that even between these images, there are variations in the image segmentation prediction. It can thus be said that segmentation is highly influenced by the brightness of the image.

False positive evaluation

A further test was conducted to determine how the model would perform on images with bushy background and no instances of the target plant present. The exposure of the image was once again varied to test how the model performs with variation in brightness. Sample results are presented in Figure 6.5.

From these images, particularly those in column three, it is clear that as the image darkens more false positives are predicted. In addition, no images containing bushy

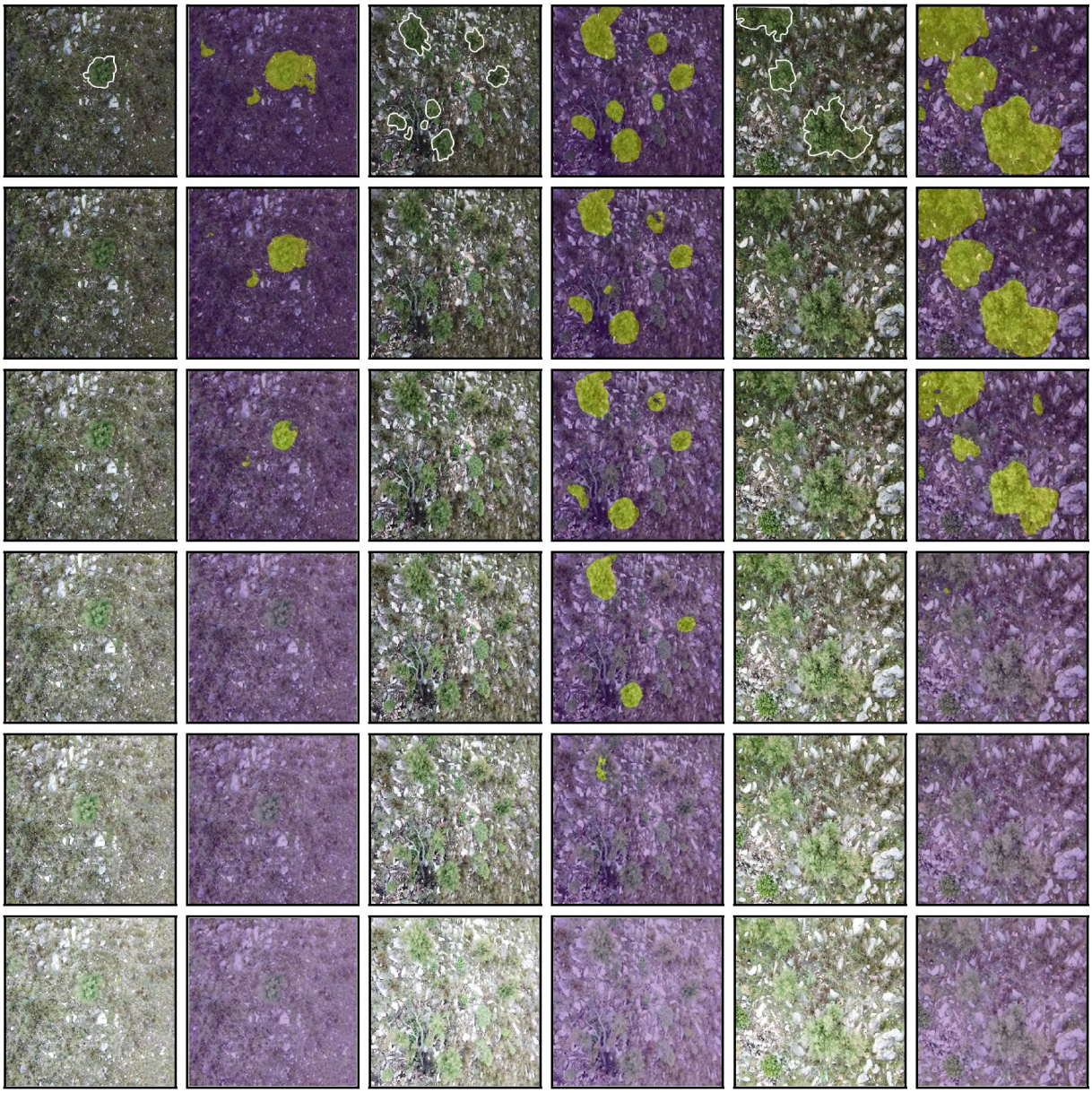


Figure 6.4: Results of varying the exposure while maintaining constant altitude and position. Exposure increases from top to bottom.

background such as these were included in the training set and as such the high number of false positives detected in the third column, which is particularly bushy, is hardly surprising.

Variation in altitude

In a similar manner to the brightness tests, the model's robustness to altitude variation was examined. The exposure of the image was kept constant, and the drone was moved

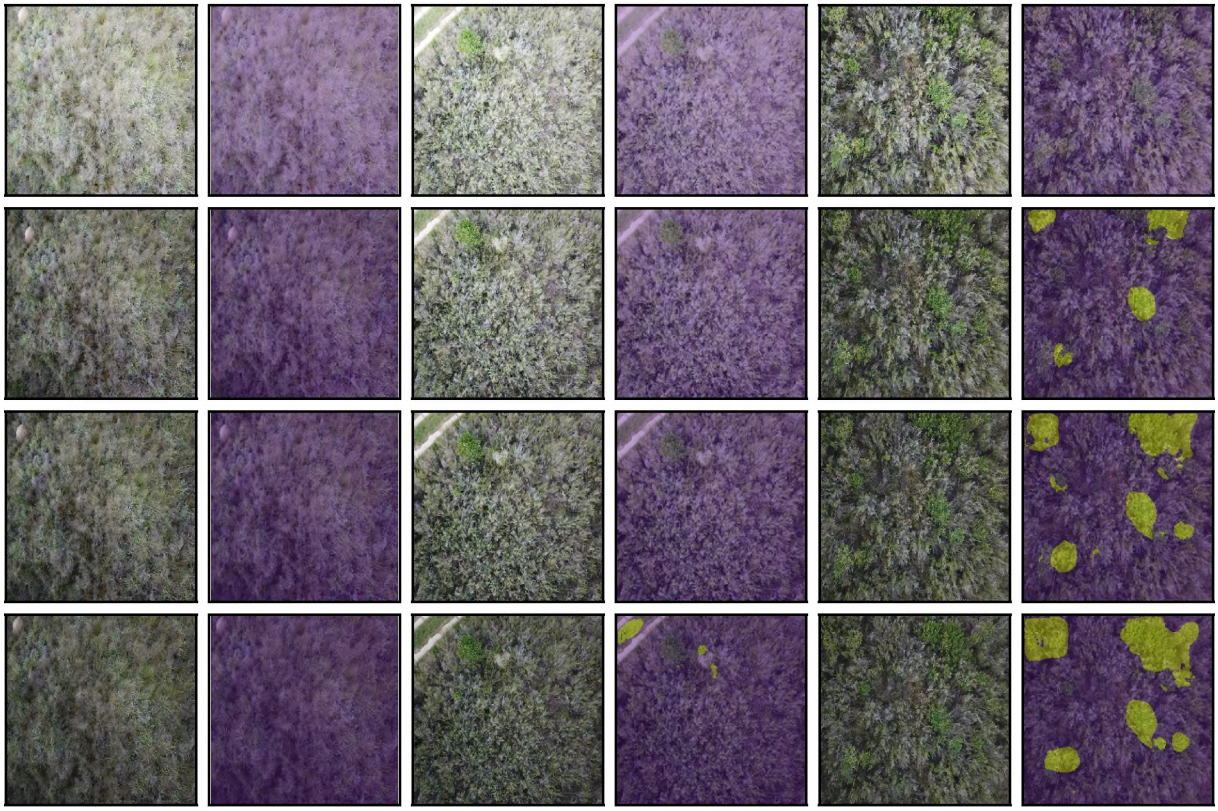


Figure 6.5: Results of varying the exposure while maintaining constant altitude and position when no target plants are present. Exposure increases from top to bottom.

vertically upwards for each test. The results of this experiment are presented in Figure 6.6.

Assessing the segmentation prediction as a function of altitude is less straight forward than assessing variation as a function of brightness, as the number of objects within the image increases with altitude. To make a fair assessment over the tested altitudes, we consider only those objects present at the lowest altitude.

In the first column, it is seen that the true positive detection is maintained until the final altitude increase, at which point it disappears. A decrease in performance past a critical altitude is also observed in column 2, where the original prediction is lost in the third row. It appears that a decrease in performance is experienced at different altitudes for the two targets, which indicates that it is likely that variation in altitude, within reasonable bounds, is dependant on the morphology of the individual target plant, but further experimentation is necessary to observe this conclusively.

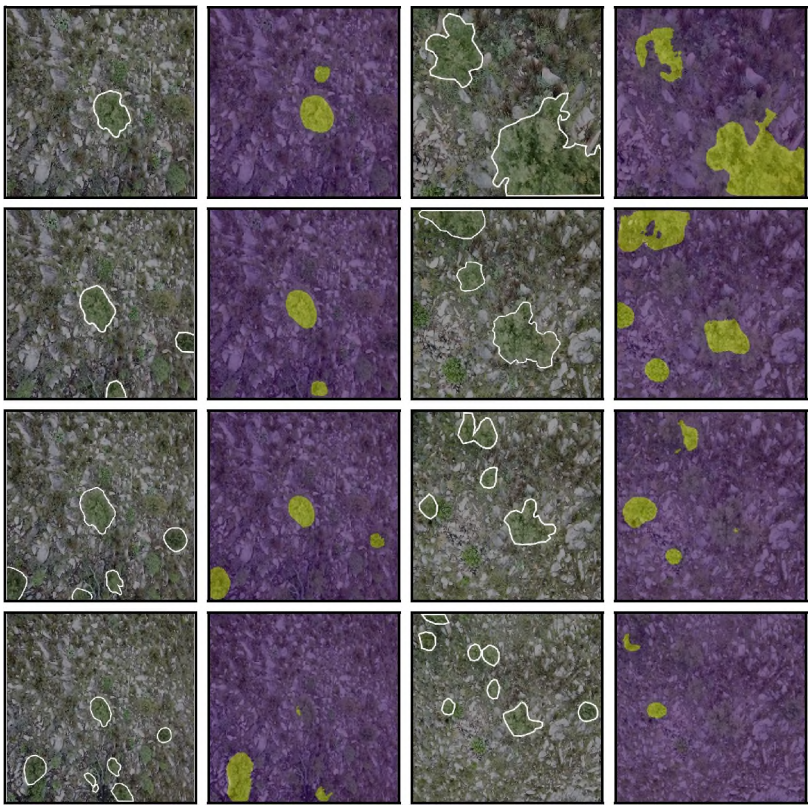


Figure 6.6: Results of varying the altitude while maintaining constant exposure and position. Altitude increases from top to bottom.

6.6.2 Execution time

Timing logs were recorded to gain an estimate of how long obtaining the bitmap from the FPV, pre-processing, segmentation and post-processing of an image takes. Each time an image was sampled from the FPV, an entry into the log was made. The average time over 41 images was calculated to be 5047.27 ± 448.58 ms. This execution time per frame is far slower than real-time, however, as this is not a time critical operation it is acceptable. The reported time does not include posting the modified canvas to the view, as the visualisation of the segmentation result could be replaced with any other function, such as a routine to spray herbicide or deliver bio-control agents, in future applications. By timing up until this point, we obtain results for the core segmentation procedure, which would be present irrespective of final use.

6.7 Summary

This chapter documented the development of a mini DJI GO application with a successfully integrated TensorFlow model, which was capable of performing inference on images sampled from the video livestream. An initial assessment of the system's performance was presented, with an average execution time in flight for image acquisition, pre-processing, classification and rendering of 5047.27 ± 448.58 ms per image. A series of experiments were performed to determine the sensitivity of the model to the image brightness, through variance of the camera's exposure, as well as sensitivity to altitude. The first and third experiments respectively indicate that mid to lower values for exposure and altitude are necessary for good predictions. However, the second experiment suggests that there may be an increase in false positive prediction at mid and lower range exposure. This means that within the optimal exposure range for segmenting the highest number of target plants successfully, a number of false positives are also likely to be predicted. This suggests room for further refinement of the model to increase its robustness to variance of brightness. However, the model did perform well when the exposure and altitude were suitable, predicting relatively few false positives and predicting the majority of the target plants as true positives, thereby confirming that this system can be used to detect the target plant, *Hakea*, in the field.

Chapter 7

Model Refinement

In the previous chapter, initial field testing of the model showed that model performance varied with brightness, performing well at some brightness levels, but poorly at others where it missed target plants and/or detected false positives. The fluctuation of model performance with brightness is not unique to our model. A study conducted by Grm *et al.* (2018) to assess the effect of varying factors related to image quality, including brightness, on model performance found that increasing the brightness of the test set resulted in a drop in performance for all models tested. The brightness of images such as those in our dataset varies depending on environmental conditions, such as the time of day and cloud cover as well as camera exposure. Therefore, the model must be robust to a reasonable degree of variation in image brightness. A possible way of achieving this is the use of brightness transformations as an additional data augmentation strategy. This technique has been employed by Guirado *et al.* (2017) in their work on detecting *Ziziphus lotus* shrubs in Google Earth™ imagery.

The dataset used to train the model in this research, containing images of target shrubs amongst other vegetation and images in which no target shrubs are present, has by its nature a high level of class imbalance, with substantially more pixels in the background class than the target class. When assessing the percentage of pixels in each class in dataset 1, it was found that only 8% of pixels belong to the target class, while the other 92% belong to the background class which consists of all landcover that is not of the target class. High levels of class imbalance can allow the dominant class to swamp loss functions such as BCE, which apply equal learning ability to all pixels and to all classes and hence, a solution optimised for only the dominant class can be reached. This has implications for model performance and as such alternative loss functions have been developed to address

this issue, including weighted BCE (WBCE), Dice loss (Milletari *et al.*, 2016) and focal loss (Lin *et al.*, 2017), amongst others.

In this chapter, we perform quantitative assessments of two complementary strategies, brightness augmentation and the use of different loss functions, to determine to what extent model performance can be improved. Qualitative analysis then provides further insight into model behaviour. Thereafter, a model is selected as the most optimal one for the task of segmenting the target shrub, *Hakea*, from other shrubs and land cover.

7.1 Data augmentation

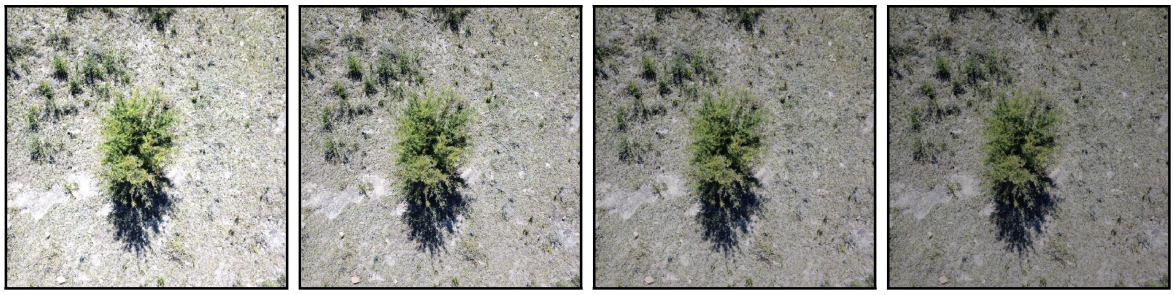
As discussed previously, U-Net was designed to make use of excessive data augmentation owing to limited available training data (Ronneberger *et al.*, 2015). When training the baseline model, augmentation functions to produce flips, rotations, shifts and zooms were performed. These are considered the standard image augmentation functions for this project, to which brightness augmentation was added.

In the HSV colour space, the hue channel contains all colour information, the saturation channel controls the intensity of the colour, and the value channel determines the colour brightness. This means that the value channel may be adjusted to change the image brightness, without affecting the colour or intensity information of the image.

This property was employed to perform brightness augmentation on the dataset to increase the model's robustness to variation in brightness. A brightness augmentation function was incorporated into the set of `ImageDataGenerator` functions, which could randomly adjust the brightness of the image such that it could be up to 30% darker or lighter than the original. A sample of images showing the results of brightness augmentation when no other augmentation is applied is shown in Figure 7.1.

With the standard set of data augmentation functions used for the baseline model, the augmentation must be applied to both the image and the mask so that when, for example, rotations, flips and zooms are performed, the mask still matches the input image. Brightness augmentation differs from other augmentation functions in this respect, as it is applied to the image, but not the mask, which is binary in nature. On this note, it is necessary to threshold the mask after augmentation such that its binary nature is preserved, as rotations and zooms may cause interpolation between pixels to occur which can result in a non-binary mask. A non-binary mask would imply a multi-class segmentation

problem, which is fundamentally different from a binary segmentation problem.



(a) Augmented (lighter) (b) Original image (c) Augmented (darker) (d) Augmented (darker)

Figure 7.1: The effect of brightness augmentation on a sample image.

7.2 Loss functions

To improve the performance of the network across all metrics, the use of a different loss function was proposed. The standard BCE loss assigned equal learning ability to each pixel in the image, whereas the loss used in the original U-Net paper (Ronneberger *et al.*, 2015) assigned a weighted loss to give greater learning ability to the edges between adjacent segmented objects, so as to distinguish individual instances. The ground truth masks created for our dataset were not created to separate individual instances of the target plant, as *Hakea* shrubs often grow into one another in dense stands. This suggests that the loss function used by Ronneberger *et al.* (2015) is not ideally suited to the task. Several alternative loss functions are introduced hereafter.

7.2.1 Loss functions to address class imbalance

WBCE

When considering that BCE is calculated across a batch as an average, it is clear that the dominant background class may dominate the contribution to the total loss. A possible way of reducing this effect is to make use of a WBCE loss, in which each class' contribution is weighted by the inverse frequency of its occurrence.

$$L_{WBCE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\omega_c g_{nc} log(p_{nc}) \tag{7.1}$$

In the case of our binary problem,

$$\omega_c = \begin{cases} 0.08 \text{ if background class} \\ 0.92 \text{ if target class} \end{cases}$$

When implemented, these weightings need to be applied to the loss in a pixel-wise fashion, which requires the creation of a weight map. This weight map may easily be computed using the binary nature of the ground truth vector, as multiplying this vector by the target class weight gives a vector with the weight value in the place of ones, while multiplying $1 - g$ by the weight for the background class gives this value where the zeros used to be. The addition of these two resulting matrices produces the final weight map, which can be generated at run-time.

In the special case where both weights are equal to one, the WBCE reduces to the standard BCE. This special case was used to verify the implementation of the WBCE function. Keras implementations of both the BCE and the WBCE with weights of [1,1] were computed for the same ground truth and predicted vectors. This yielded results of BCE = 0.039709575 and WBCE (w=[1,1]) = 0.039709594, which are equal to seven decimal places.

Focal loss

Another loss function that attempts to reduce the negative effects of class imbalance on performance is the focal loss, proposed by Lin *et al.* (2017). Within the dataset, some samples are easy to predict and these predictions obtain a high prediction probability, while other samples are difficult and obtain a low probability. Focal loss exploits this variation in prediction confidence by introducing a modulating factor, which reduces a sample's contribution to the total loss when its prediction probability is high. This increases the contribution to the loss of pixels that are misclassified.

Focal loss is defined in Equation 7.2, where γ is an additional hyper-parameter introduced which can be tuned, although Lin *et al.* (2017) found $\gamma = 2$ to work well. Like BCE, the average across all samples in the batch is calculated.

$$L_{Focal} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}(1 - p_t)^{\gamma} g_{nc} log(p_{nc}) \tag{7.2}$$

Weighted focal loss

In the same manner that a WBCE is achieved, a weighted focal (WF) loss can be defined as in Equation 7.3, which Lin *et al.* (2017) found to give improved performance over the non-weighted focal loss.

$$L_{\alpha_c Focal} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\alpha_c g_{nc} log(p_{nc}) \tag{7.3}$$

Dice loss

The Dice loss, introduced by Milletari *et al.* (2016), is given by Equation 7.4, where the second term is the Dice coefficient and ϵ is used to avoid divide by zero issues due to empty sets, which is accepted practice as per Sudre *et al.* (2017). Divide by zero issues occur when there are no positive pixels within the mask, nor in the predicted output – that is, perfect true negative prediction.

$$L_{Dice}(g_i, p_i) = 1 - \frac{2\sum_i^N p_i g_i + \epsilon}{\sum_i^N p_i^2 + \sum_i^N g_i^2 + \epsilon} \tag{7.4}$$

Intersection over union loss

The intersection over union (IOU) score, or Jaccard Similarity Coefficient, is given as the intersection of the two sets, $G \cap P$, over the union of the two sets, $G \cup P$. An approximate IOU, using the probability values, from Rahman & Wang (2016) is given as follows, where ϵ is again used to avoid the divide by zero error due to empty sets.

$$IOU(g_i, p_i) = \frac{G \cap P}{G \cup P} = \frac{|G \cap P|}{|A| + |B| - |G \cap P|} = \frac{\sum_i^N g_i p_i}{\sum_i^N (g_i + p_i - g_i p_i)} \tag{7.5}$$

It follows that the IOU loss is as given in Equation 7.6.

$$L_{IOU}(g_i, p_i) = 1 - \frac{\sum_i^N g_i p_i + \epsilon}{\sum_i^N (g_i + p_i - g_i p_i) + \epsilon} \tag{7.6}$$

7.2.2 Loss functions to address uncertainty in edge annotations

Supervised training of deep CNNs makes it necessary to have a set of training images and a corresponding annotation mask for each image in the training set. These masks distinguish which areas of the image belong to what class and are usually created by hand. When creating an annotation mask for an object with distinct edges, such as a car, a book or a person, it is relatively easy to distinguish which parts of the image belong to which class. On the other hand, for objects with indistinct edges, such as vegetation, it is harder to create an annotation mask, and there is some level of human subjectivity introduced as annotators must use their discretion at the edges of objects as to which class the object belongs, as demonstrated in Figure 7.2. This results in ground truth annotations of low confidence within these indistinct regions.

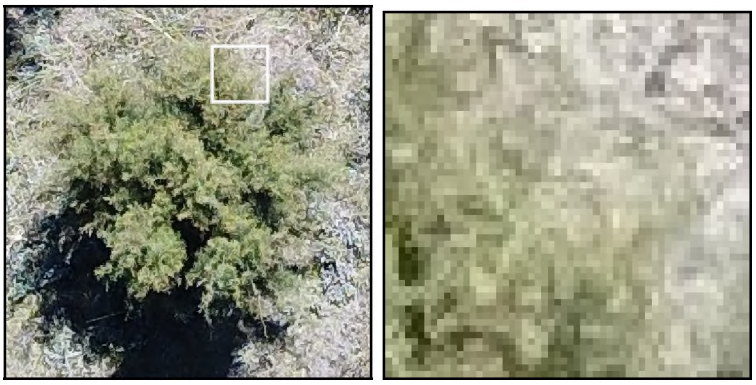


Figure 7.2: (Left) A single target object extracted from a larger image. (Right) Area demarcated by a white square in the image on the left enlarged to demonstrate the indistinct nature of the target object's edges.

While the loss functions in the previous section apply different learning ability to pixels of different classes or segmentation confidence, none of these functions takes annotation confidence into account. This implies that pixels of dubious class within indistinct edges contribute the same amount to the loss as a pixel of confident class, found away from the edges. This suggests a need for a loss function that takes into account the low confidence of annotation of pixels within indistinct edges. There is little mention of dealing with low confidence in annotation in the literature. One method mentioned for addressing errors in annotation at the edges of objects is the use of the morphological erosion operation to erode the object edges and replace these pixels with others from the background class (Bischke *et al.*, 2018).

The widely used WBCE function, which weights the contributions of pixels according to their inverse class frequency, displays a common use of a weight map in loss functions.

This weight map can be implicitly created within the loss function itself using tensor transformations. However, the approach of using a more complex, precomputed weight map is attributed to Ronneberger *et al.* (2015), who made use of a weight map to force a network to learn the spaces between cells in a biomedical application. The weight map designed by Ronneberger *et al.* (2015) up-weighted the contribution to the loss of pixels that fell within these regions in addition to addressing class frequency imbalance. Although the use of weight maps as part of the loss functions mentioned here do not address low confidence in annotation, they do indicate that it is possible to adjust the contributions of pixels that meet a specified criterion to the loss. This indicates that there may be potential in the utilization of weight maps to down-weight the loss of pixels within indistinct edges.

Weight map based loss definition

The definition for BCE was extended to include an additional term ω_{map}, a pre-computed pixel-wise weight map, as shown in Equation 7.7,

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\omega_{map}g_{nc}log(p_{nc}) \tag{7.7}$$

where ω_{map} is defined with two additional hyper-parameters α and β as,

$$\omega_{map} = \begin{cases} \alpha \text{ if pixel falls within edge of thickness } t \\ \beta \text{ otherwise.} \end{cases}$$

In the same manner as BCE was extended to give class WBCE, the definition for weight map based loss is extended to give a class balanced version in Equation 7.8.

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C}\omega_{map}\omega_{c}g_{nc}log(p_{nc}) \tag{7.8}$$

The weight maps are pre-computed for each image before being passed via the network to the loss function. To compute each weight map, the contour of the segmentation mask is found using **OpenCV**'s *findContour* function. All values within the contour of thickness t are set to value α, while all other values are set to β. Thus there are three hyper-parameters, t, α and β, which can be experimentally adjusted to find the best weight

map design for the problem.

7.3 Training

In this section, necessary information pertaining to the configuration of the model, training scheme and evaluation is given.

7.3.1 U-Net configuration

The same configuration as for the baseline model was used, with the addition of brightness augmentation to the set of data generator functions when appropriate.

7.3.2 Training scheme

Experimentation was performed to examine the performance of the models, both quantitatively and qualitatively. Three individual experiments were carried out; the first to determine whether brightness augmentation improved the performance of the model, the second to determine whether class imbalance can be accounted for on this dataset using a loss function and finally, the third experiment, to determine whether the uncertainty in edge annotation can be accounted for using a weight map based loss.

In Experiment 1, two models were trained using BCE loss, with the first trained using the standard chosen set of data augmentation functions only and the other trained using brightness augmentation in addition to the standard augmentation functions. Experiment 2 made use of the loss functions defined in Section 7.2.1, while Experiment 3 made use of the weight map based loss defined in Equation 7.8. A series of different combinations of the hyper-parameters α, β and t were used to train the model, in particular, each weight combination w of set $W = \{1,0;\ 1,0.2;\ 1,0.5;\ 2,0.5;\ 5,1\}$ along with each thickness t of set $T = \{1;3;5\}$. In all three experiments, k-fold validation was first performed to gauge performance on dataset 1 and thereafter the models were trained on dataset 1 and tested on the unseen dataset 2. Based on the findings of Experiment 1, Experiments 2 and 3 both made use of brightness augmentation.

The training scheme made use of a batch size of two and both batchwise and average metrics. Batchwise metrics are calculated over all samples in a batch irrespective of

which image they belong to, while average metrics are calculated per image and then averaged across all images in the batch. Calculating both sets of metrics gives greater insight into the behaviour of the model.

7.3.3 Metrics

High levels of class imbalance can result in the accuracy metric giving an artificially high indication of performance, as even if every image is predicted to contain only pixels belonging to the background class, an accuracy of 92%, which is the frequency of occurrence of the dominant class, can be achieved. This indicates the importance of metrics that reflect the performance of predictions in the positive class, such as recall, for a reflection of the true positives predicted. A high recall value is crucial to detecting target plants, while a high precision value is necessary to ensure that the high recall value is not achieved simply by predicting many false positives. In the case of our application, a moderate precision value is acceptable, whereas a moderate recall value is not. In addition to the precision, recall, F1 score and accuracy metrics used before, we use the Dice coefficient and IOU score, introduced in Section 7.2.1, to gain further insight into model performance.

When segmenting an image for a target plant, it is entirely possible that the plant may not be present and as such, the entire image would contain no true positives. Many metrics, such as precision, recall, F1 score, IOU and the Dice coefficient, expect there to be true positives and/or false positives predicted. When there are no positives present, and all pixels are predicted as true negatives, these metrics encounter a divide by zero error. Possible ways of dealing with this include the removal of these meaningless values from the array and the use of a very small number, epsilon, to be included in the denominator to prevent the divide by zero issues and provide a score of zero.

Removing perfectly predicted true negative images from the metric calculation for the affected metrics makes sense when calculating metrics for assessment, as these metrics do not give any indication of perfect true negative performance. On the other hand, setting the score to zero through the use of epsilon in the denominator has larger ramifications, as a metric score of zero implies a loss of one, in other words, a completely incorrect segmentation. Adding this value to a metric would significantly lower the average metric over all images. A third alternative provides a solution to this, adding epsilon to both the numerator and denominator, as seen in the definition for generalized Dice loss in (Sudre *et al.*, 2017). This means that when perfect true negatives are predicted we obtain epsilon over epsilon and the metric score evaluates to one, resulting in a loss of zero, as it should

be. However, this third approach is only suitable for the loss function, as a metric of score one for a perfectly predicted true negative image makes no sense for a metric like recall, which deals only with positive predictions. Hence, by removing perfectly predicted true positives from arrays of metrics and adding epsilon to both numerator and denominator when calculating loss, the divide by zero errors encountered by metrics pertaining to positive prediction can be mitigated.

7.4 Performance analysis

The two datasets used have distinctly different brightness levels, having been collected on different days under different environmental conditions, whereas the brightness within each dataset is relatively constant. K-fold validation was performed using dataset 1. When training for inference in these experiments, the models were trained on dataset 1 and tested on dataset 2, which was less bright than the former dataset. It is necessary that a model trained on dataset 1 should be able to generalise its performance to dataset 2 and hence that the k-fold performance be comparable to test inference performance.

7.4.1 Brightness augmentation

Table 7.1: K-fold metrics – brightness augmentation

Batchwise	Precision	Recall	F1	Dice	IOU	Accuracy
Baseline BCE	0.87±0.05	0.56±0.22	0.72±0.09	0.65±0.23	0.51±0.20	0.96±0.01
Brightness	0.88±0.06	0.43±0.31	0.63±0.21	0.50±0.32	0.39±0.27	0.95±0.02
Average	Precision	Recall	F1	Dice	IOU	
Baseline BCE	0.70±0.14	0.56±0.21	0.72±0.06	0.49±0.21	0.41±0.18	
Brightness	0.72±0.12	0.43±0.29	0.62±0.21	0.39±0.27	0.32±0.24	

When considering the k-fold results in Table 7.1, it is clear that on average, the non-augmented baseline model outperforms the brightness augmented one by approximately 10%. This is unsurprising, as all images in both the training and validation folds for the baseline model have the same brightness level, whereas when brightness augmentation is applied, while the validation fold does not experience a brightness fluctuation, the training folds do. When considering the upper bound of the standard deviation from the mean for the augmented model, it is seen that the model can achieve a higher performance than the baseline, even if not on average. This could possibly be due to random brightness augmentations that happen to tune the model to better cope with the slight

fluctuations within dataset 1. The case where the model achieved the lower bound of the standard deviation of the mean could be a case where all augmentations happened to lower the brightness in the training fold, which reduced the accuracy on the bright validation fold.

Table 7.2: Performance of the brightness augmented inference model on the unseen test set

Batchwise	Precision	Recall	F1	Dice	IOU	Accuracy
Baseline BCE	0.93	0.28	0.43	0.43	0.27	0.93
Brightness	0.83	0.69	0.75	0.75	0.61	0.96
Average	Precision	Recall	F1	Dice	IOU	
Baseline BCE	0.80	0.29	0.44	0.33	0.25	
Brightness	0.65	0.65	0.78	0.52	0.44	

The results of inference on the test set in Table 7.2 give insight into how well the model generalises to unseen data of different brightness levels to the training set. It is seen that the baseline model does not maintain its performance on unseen data, dropping about 30% of its performance. In contrast, the augmented model obtains a score on the test set within its standard deviation of the mean from k-fold validation, performing better than the average k-fold score. These observations indicate that a more robust model can be obtained through the use of brightness augmentation.

Qualitative assessment of the two models visualised in Figure 7.3, one trained with brightness augmentation (Aug) and the other without (No Aug), confirms that brightness augmentation is beneficial to model performance. With augmentation, a far better true positive detection rate is seen, with all six instances of the target shrub detected well, and the majority of the objects' areas detected. The augmented model detected a few more false positives than the unaugmented model, but the trade-off was a beneficial one to overall performance. These observations agree with the pixel-wise metrics.

7.4.2 Class imbalance

As Experiment 1 showed that using brightness augmentation allowed sufficient generalisation when trained on dataset 1 to perform inference at a comparable level of performance on dataset 2, we take it that any substantial differences in performance between the k-fold validation results and the test set inference results are due to the behaviour of the loss function. When referring to the BCE loss in this and the next experiments, it is the BCE loss with brightness augmentation to which the reference is intended.

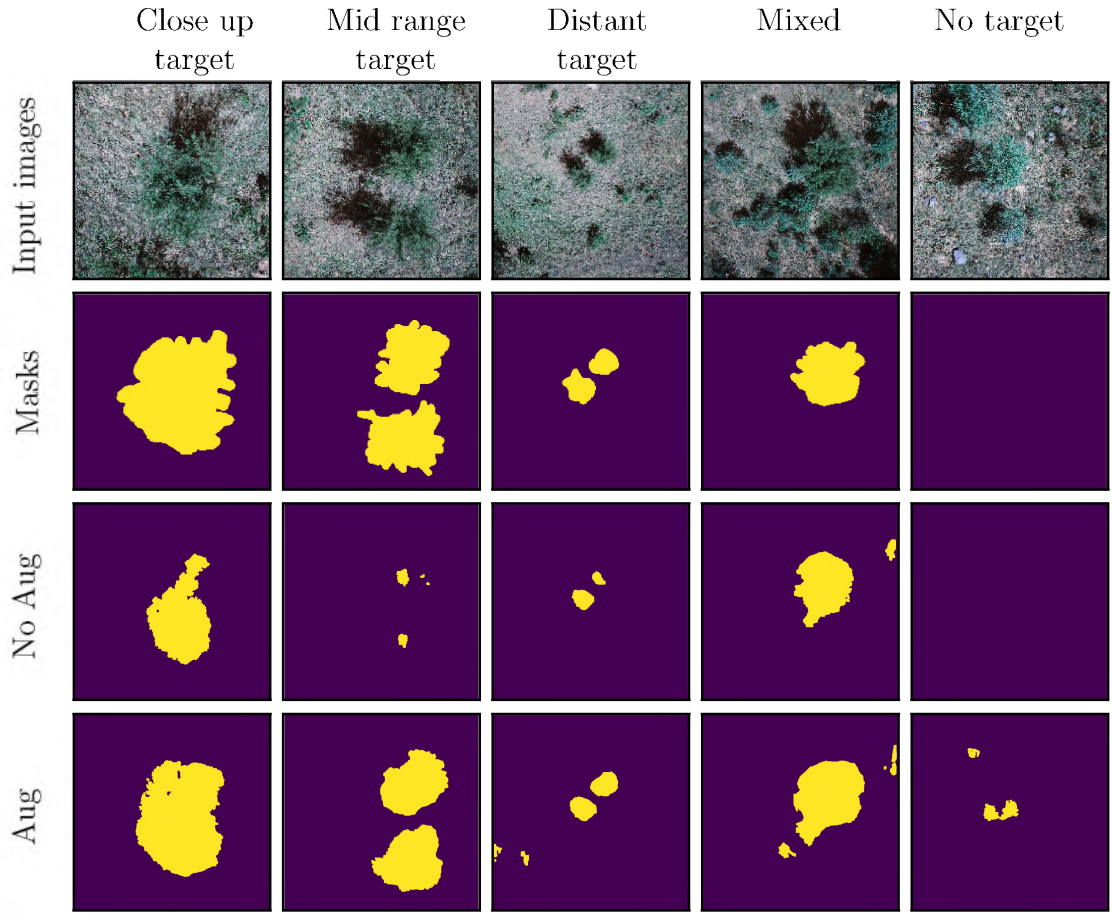


Figure 7.3: A selection of input images for different cases of image composition, along with their binary masks and the predicted segmentation for top-performing models (No Aug and Aug). Used for brightness augmentation comparison.

The k-fold performance of models trained with loss functions to counter class imbalance (CCI models) largely show an improvement in Table 7.3 when compared to BCE loss in Table 7.1, with only focal loss with $\gamma = 2$ performing less well on the average metrics. This strongly indicates that class imbalance has a large impact on the performance of our models, but that this can be improved upon through the use of an appropriate loss function.

It is interesting to note that accounting for class imbalance decreases the standard deviation caused by brightness variance, that was present across folds when using BCE. When considering that these loss functions focus more strongly on positive target samples or samples which are more difficult to classify than background samples, it can perhaps be conjectured that this focus better allows the model to learn invariance to fluctuations within the target class, with lesser distraction by brightness variance in the background class.

On the whole, the models generalise well to the unseen data, as seen in the results in Table 7.4. For the batchwise metrics, the last four losses fall within the standard deviation from the mean from k-fold, with only the Dice loss more than 2% beyond this threshold. For the average metrics, the final three losses, as well as the focal loss with $\gamma = 0.5$, do not fall within the standard deviation of the mean calculated during k-fold validation. Again the Dice loss is the worst of these, scoring 14% below the minimum expected deviation, while focal loss with $\gamma = 0.5$ is 5% below this threshold, IOU loss 4% below it and the WF loss with $\gamma = 0.5$ scoring only 1% better than its maximum threshold. The strongest candidates in terms of batchwise metrics are the WF loss with $\gamma = 1$ and $\gamma = 0.5$ on the inference results, while the top performers in terms of the average metrics are the weighted and unweighted focal loss with $\gamma = 0.5$ and WBCE. It is noted that the models that do not obtain test scores within the k-fold validation standard deviations are not necessarily poorly performing models, only that their performance is inconsistent across the different datasets.

When trying to determine which model is the best for our task, we first consider the average metrics from k-fold validation and exclude all models achieving less than 70% for their F1 score. This removes all focal loss as well as WF loss with $\gamma = 2$. We then consider the results from the test set. From this table and the remaining models not eliminated previously, we select the top 3 performers in F1 score, in order, WF loss with $\gamma = 0.5$, WBCE and WF loss with $\gamma = 1$. These are also the best batchwise performers. Dice loss

Table 7.3: K-fold metrics – class balancing loss functions

Batchwise metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WBCE		0.53±0.10	0.93±0.06	0.67±0.10	0.67±0.10	0.51±0.10	0.93±0.03
Focal Loss	$\gamma = 2$	0.65±0.28	0.44±0.33	0.49±0.32	0.49±0.32	0.39±0.30	0.95±0.02
	$\gamma = 1$	0.82±0.17	0.51±0.23	0.66±0.15	0.60±0.24	0.46±0.22	0.96±0.02
	$\gamma = 0.5$	0.78±0.27	0.53±0.21	0.70±0.08	0.63±0.22	0.49±0.19	0.96±0.01
W Focal	$\gamma = 2$	0.47±0.14	0.83±0.07	0.59±0.14	0.59±0.14	0.43±0.13	0.91±0.03
	$\gamma = 1$	0.51±0.16	0.85±0.06	0.63±0.15	0.63±0.15	0.47±0.14	0.92±0.04
	$\gamma = 0.5$	0.50±0.11	0.91±0.07	0.64±0.10	0.64±0.10	0.48±0.11	0.92±0.03
Dice Loss		0.64±0.16	0.86±0.04	0.72±0.12	0.72±0.12	0.57±0.14	0.95±0.02
IOU Loss		0.55±0.13	0.86±0.06	0.66±0.09	0.66±0.09	0.50±0.11	0.94±0.02
Average metrics		Precision	Recall	F1	Dice	IOU	
WBCE		0.46±0.09	0.92±0.06	0.77±0.06	0.52±0.11	0.43±0.09	
Focal Loss	$\gamma = 2$	0.55±0.19	0.44±0.31	0.51±0.29	0.37±0.26	0.31±0.24	
	$\gamma = 1$	0.68±0.15	0.51±0.24	0.67±0.15	0.45±0.21	0.36±0.19	
	$\gamma = 0.5$	0.63±0.25	0.52±0.20	0.69±0.06	0.47±0.21	0.38±0.19	
W Focal	$\gamma = 2$	0.42±0.10	0.82±0.08	0.69±0.11	0.46±0.09	0.36±0.08	
	$\gamma = 1$	0.44±0.15	0.85±0.06	0.73±0.09	0.47±0.15	0.39±0.13	
	$\gamma = 0.5$	0.44±0.11	0.89±0.08	0.75±0.07	0.49±0.12	0.40±0.11	
Dice Loss		0.58±0.12	0.84±0.06	0.82±0.05	0.57±0.12	0.49±0.10	
IOU Loss		0.53±0.13	0.84±0.07	0.80±0.06	0.52±0.13	0.45±0.11	

Table 7.4: Performance of the inference models for different loss functions on the unseen test set

Batchwise metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WBCE		0.56	0.92	0.70	0.70	0.54	0.93
Focal Loss	$\gamma = 2$	0.93	0.30	0.46	0.46	0.30	0.93
	$\gamma = 1$	0.72	0.75	0.74	0.74	0.58	0.95
	$\gamma = 0.5$	0.73	0.75	0.74	0.74	0.58	0.95
W Focal	$\gamma = 2$	0.54	0.93	0.68	0.68	0.52	0.92
	$\gamma = 1$	0.88	0.74	0.80	0.80	0.67	0.97
	$\gamma = 0.5$	0.67	0.89	0.76	0.76	0.62	0.95
Dice Loss		0.34	0.92	0.50	0.50	0.33	0.83
IOU Loss		0.51	0.61	0.56	0.56	0.38	0.91
Average metrics		Precision	Recall	F1	Dice	IOU	
WBCE		0.44	0.94	0.80	0.49	0.42	
Focal Loss	$\gamma = 2$	0.87	0.32	0.47	0.38	0.28	
	$\gamma = 1$	0.57	0.66	0.71	0.47	0.39	
	$\gamma = 0.5$	0.61	0.71	0.80	0.53	0.45	
W Focal	$\gamma = 2$	0.40	0.95	0.76	0.46	0.38	
	$\gamma = 1$	0.67	0.76	0.79	0.59	0.51	
	$\gamma = 0.5$	0.49	0.91	0.83	0.52	0.45	
Dice Loss		0.32	0.91	0.63	0.38	0.30	
IOU Loss		0.48	0.64	0.70	0.41	0.33	

is notably excluded, as although it obtained top performance during k-fold validation, its performance plummeted on the test set, indicating that its performance was very specific to the first dataset and perhaps overfit to it.

When considering the qualitative performance of the top three CCI models (see Figure 7.4), it is immediately obvious that the trade-off between precision and recall is once again apparent. WBCE and WF $\gamma = 0.5$ both exhibit a shift towards better recall, with objects detected more fully, along with a corresponding increase in loss of precision and hence, a greater number of detected false positives. It is suggested that WF loss is better than WBCE in this regard, predicting fewer FPs while having a similar recall, particularly when considering the fifth column in which no target is present. The $\gamma = 1$ model depicts a shift in the other direction, with better precision and fewer false positives (none at all in column 5) while targets are less fully detected, indicating a lower pixel-wise recall. However, as all targets are detected reasonably well, this model is selected as the top-performing CCI model due to its minimal false positives. This is further supported by visual examination of the model's performance across all test images. It is interesting to note that this was the model that achieved the best batchwise F1 score, but not the best average score.

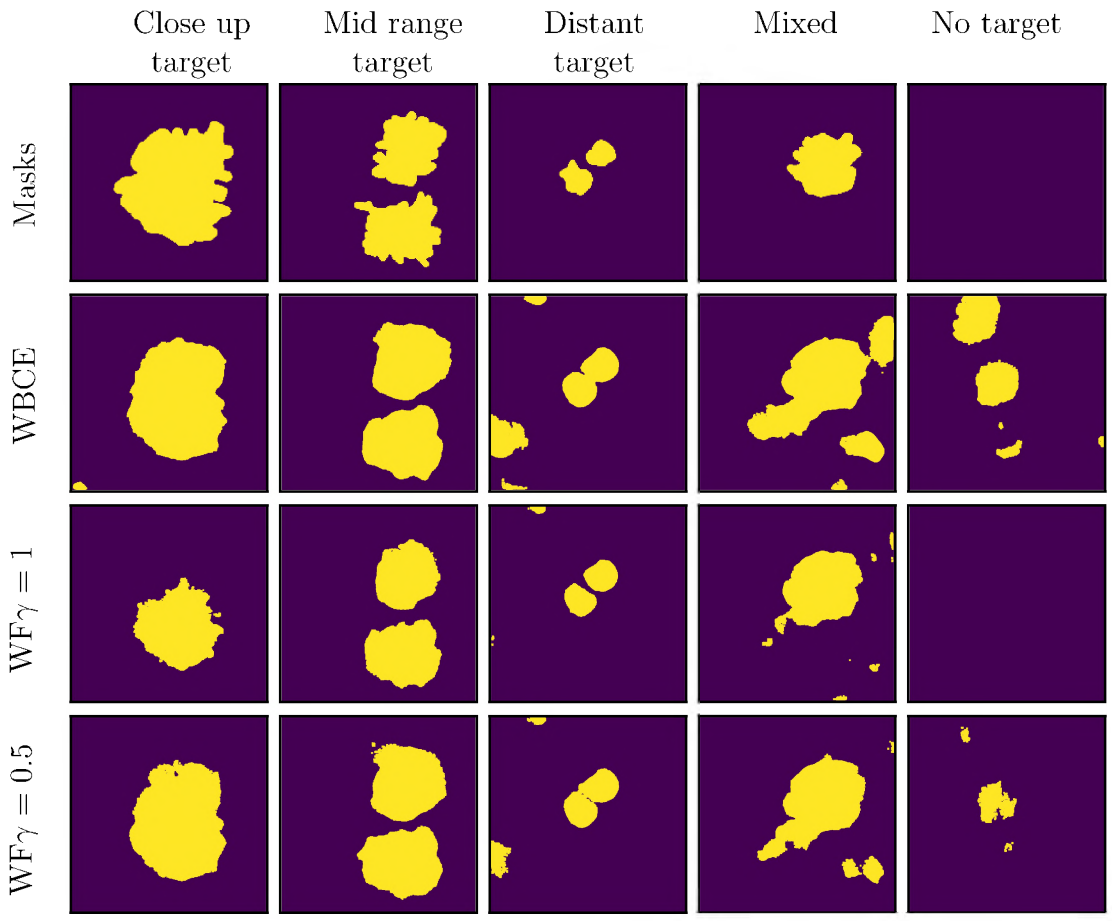


Figure 7.4: A selection of input images for different cases of image composition, along with their binary masks and the predicted segmentation for top-performing models. Used for class imbalance comparison.

7.4.3 Uncertainty in edge annotation

In this experiment, the weight map based loss defined in Equation 7.8 was applied. A distinction is made between WBCE, which makes use of class weights, and the weight map based loss, which uses weight maps to weight individual pixels.

Table 7.5: K-fold metrics for weight map design with varying weights and thicknesses

Batchwise metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WBCE		0.53±0.10	0.93±0.06	0.67±0.10	0.67±0.10	0.51±0.10	0.93±0.03
t=1	w=1,0	0.49±0.16	0.91±0.06	0.63±0.14	0.63±0.14	0.48±0.15	0.92±0.04
	w=1,0.2	0.44±0.13	0.89±0.04	0.57±0.14	0.57±0.14	0.41±0.12	0.89±0.04
	w=1,0.5	0.45±0.10	0.91±0.06	0.60±0.10	0.60±0.10	0.44±0.10	0.91±0.03
	w=2,0.5	0.51±0.11	0.93±0.04	0.65±0.10	0.65±0.10	0.49±0.11	0.93±0.02
	w=5,1	0.51±0.08	0.94±0.03	0.66±0.07	0.66±0.07	0.49±0.08	0.92±0.08
t=3	w=1,0	0.50±0.13	0.91±0.05	0.64±0.13	0.64±0.13	0.48±0.13	0.92±0.03
	w=1,0.2	0.53±0.08	0.92±0.06	0.67±0.08	0.67±0.08	0.51±0.08	0.93±0.02
	w=1,0.5	0.49±0.12	0.91±0.06	0.62±0.12	0.62±0.12	0.46±0.11	0.92±0.02
	w=2,0.5	0.50±0.13	0.93±0.03	0.64±0.13	0.64±0.13	0.49±0.12	0.92±0.03
	w=5,1	0.47±0.12	0.94±0.02	0.62±0.12	0.62±0.12	0.46±0.11	0.91±0.02
t=5	w=1,0	0.55±0.11	0.87±0.06	0.67±0.10	0.67±0.10	0.51±0.11	0.94±0.02
	w=1,0.2	0.54±0.11	0.90±0.05	0.67±0.10	0.67±0.10	0.51±0.10	0.93±0.03
	w=1,0.5	0.55±0.09	0.90±0.04	0.68±0.07	0.68±0.07	0.51±0.08	0.93±0.03
	w=2,0.5	0.51±0.11	0.91±0.05	0.64±0.09	0.64±0.09	0.48±0.09	0.93±0.02
	w=5,1	0.46±0.15	0.93±0.03	0.60±0.16	0.60±0.16	0.45±0.15	0.90±0.04
Average metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WBCE		0.46±0.09	0.92±0.06	0.77±0.06	0.52±0.11	0.43±0.09	
t=1	w=1,0	0.46±0.12	0.90±0.07	0.76±0.07	0.51±0.12	0.42±0.11	
	w=1,0.2	0.41±0.11	0.87±0.06	0.71±0.08	0.46±0.11	0.37±0.10	
	w=1,0.5	0.41±0.09	0.90±0.06	0.73±0.07	0.47±0.09	0.38±0.07	
	w=2,0.5	0.44±0.11	0.92±0.04	0.77±0.05	0.49±0.12	0.40±0.10	
	w=5,1	0.44±0.10	0.93±0.04	0.76±0.04	0.50±0.12	0.41±0.09	
t=3	w=1,0	0.43±0.09	0.89±0.06	0.74±0.08	0.48±0.10	0.39±0.08	
	w=1,0.2	0.46±0.10	0.90±0.07	0.78±0.05	0.51±0.11	0.42±0.09	
	w=1,0.5	0.44±0.09	0.90±0.06	0.75±0.05	0.49±0.10	0.40±0.08	
	w=2,0.5	0.43±0.11	0.92±0.05	0.75±0.06	0.48±0.12	0.40±0.10	
	w=5,1	0.43±0.11	0.93±0.03	0.76±0.08	0.49±0.12	0.41±0.10	
t=5	w=1,0	0.50±0.09	0.86±0.07	0.78±0.03	0.53±0.11	0.44±0.10	
	w=1,0.2	0.46±0.08	0.88±0.07	0.77±0.06	0.50±0.10	0.42±0.08	
	w=1,0.5	0.47±0.10	0.89±0.05	0.77±0.06	0.51±0.12	0.42±0.09	
	w=2,0.5	0.46±0.10	0.90±0.05	0.77±0.06	0.50±0.11	0.42±0.09	
	w=5,1	0.44±0.14	0.92±0.04	0.76±0.09	0.49±0.15	0.41±0.13	

There is minimal variation in performance between the different hyper-parameter configurations in Table 7.5, with all models achieving between 0.57 and 0.68 on the batchwise metrics (with all bar w=1,0.2 t=1 achieving over 60%) and all achieving between 71% and 78% for the average metrics. If we consider the WBCE as a baseline, we observe a 1% average improvement by both the w=1,0.2 t=3 and w=1,0 t=5 models, which additionally have a slightly lower standard deviation than WBCE.

Table 7.6: Performance of the inference models for varying weight map design on the unseen test set

Batchwise metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WBCE		0.56	0.92	0.70	0.70	0.54	0.93
t=1	w=1,0	0.67	0.83	0.74	0.74	0.59	0.95
	w=1,0.2	0.68	0.86	0.76	0.76	0.61	0.95
	w=1,0.5	0.52	0.93	0.67	0.67	0.50	0.92
	w=2,0.5	0.56	0.88	0.69	0.69	0.52	0.93
	w=5,1	0.78	0.65	0.71	0.71	0.55	0.95
t=3	w=1,0	0.79	0.71	0.75	0.75	0.60	0.96
	w=1,0.2	0.82	0.68	0.74	0.74	0.59	0.96
	w=1,0.5	0.40	0.95	0.57	0.57	0.40	0.87
	w=2,0.5	0.65	0.83	0.73	0.73	0.57	0.94
	w=5,1	0.36	0.88	0.51	0.51	0.34	0.84
t=5	w=1,0	0.22	0.97	0.36	0.36	0.22	0.68
	w=1,0.2	0.79	0.76	0.78	0.78	0.63	0.96
	w=1,0.5	0.78	0.85	0.81	0.81	0.69	0.96
	w=2,0.5	0.81	0.80	0.80	0.80	0.67	0.96
	w=5,1	0.70	0.73	0.71	0.71	0.55	0.95
Average metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WBCE		0.44	0.94	0.80	0.49	0.42	
t=1	w=1,0	0.55	0.88	0.83	0.55	0.48	
	w=1,0.2	0.52	0.88	0.85	0.53	0.47	
	w=1,0.5	0.41	0.94	0.77	0.46	0.39	
	w=2,0.5	0.47	0.85	0.81	0.48	0.41	
	w=5,1	0.65	0.61	0.75	0.51	0.42	
t=3	w=1,0	0.67	0.68	0.78	0.55	0.47	
	w=1,0.2	0.68	0.68	0.78	0.55	0.47	
	w=1,0.5	0.34	0.96	0.67	0.40	0.33	
	w=2,0.5	0.55	0.85	0.83	0.54	0.47	
	w=5,1	0.31	0.86	0.60	0.36	0.28	
t=5	w=1,0	0.21	0.98	0.49	0.29	0.21	
	w=1,0.2	0.58	0.73	0.82	0.52	0.44	
	w=1,0.5	0.58	0.84	0.80	0.58	0.50	
	w=2,0.5	0.59	0.81	0.83	0.56	0.48	
	w=5,1	0.62	0.73	0.81	0.54	0.47	

Substantially more fluctuation in performance is visible when considering the test results in Table 7.6, with a range in batchwise F1 score of 0.36 to 0.81. It is interesting to note that the worst performance was achieved by w=1,0 t=5, one of the models that performed best on the k-fold validation. The range of performance of the average metrics was 0.49 to 0.85, again with w=1,0 t=5 performing least well. A number of models performed better than WBCE on the test set, with the batchwise score being beaten by 11% and the average score by 5%. This suggests that the weight map based loss can result in models which generalise better than models trained with equal contributions from edge pixels as others. However, considering the poor performance of w=1,0 t=5 on the test set, in which edge pixels make no contribution to the loss, it is seen that edge pixels do have a role to

play in the loss, albeit a down-weighted one.

Considering that the performance of each of the models was very similar for k-fold validation, the most optimal models are selected mainly based on performance on the test set, for further qualitative analysis. The only model which is excluded based on k-fold results is w=1,0.2 t=1, due to its sub 60% k-fold batchwise performance. The results on the inference model lead us to immediately exclude the w=1,0 t=5 model, as it achieves a significantly lower F1 score than the remaining models. The model w=1,0.2 t=1 achieved the top average score with models w=1,0 t=1, w=2,0.5 t=3 and w=2,0.5 t=5 achieving equal second position. The top batchwise performers interestingly all had thickness 5, w=1,0.2, w=1,0.5 and w=2,0.5, which also achieved a top average performance. We consider all six of these for qualitative analysis, with w=1,0.2 t=1 included for interest due to its strong test performance despite its exclusion due to poor k-fold performance.

From qualitative analysis of these models, visualised in Figure 7.5, we observe than w=1,0 t=1 and w=2,0.5 t=3 both achieve excellent object-level recall with a corresponding very high false positive rate. The segmentation results were not all that different from those of w=1,0.2 t=1. Models w=2,0.5, w=1,0.2 and w=1,0.5 all with t=5 show similar performance, with high object-level recall and substantially fewer false positives than the first three models mentioned, suggesting that excluding a wider band of edge pixels causes a reduction in false positives. Of these three models, w=1,0.2 and w=2,0.5 both appear to have slightly fewer false positives than w=1,0.5. In addition, these two models also have an almost completely opposite false positive distribution. This makes it difficult to determine which of these models is the most optimal, but as the w=2,0.5 model achieves top three scores for F1 for both k-fold and inference pixel-level metrics, this was selected as the optimal model.

7.4.4 Comparison of top models from qualitative assessment

In this section, we consider the top-performing models from the previous two sections and examine them in greater detail to determine which is the most optimal model for use in the following chapter for field testing.

When we consider the generalisation of the model to the unseen test set, the most optimal model is one that a) maintains its performance from k-fold validation when applied to the test set, within a standard deviation of the mean and b) where the model offers the best recall possible with a strong precision, but that this is in terms of both pixel and

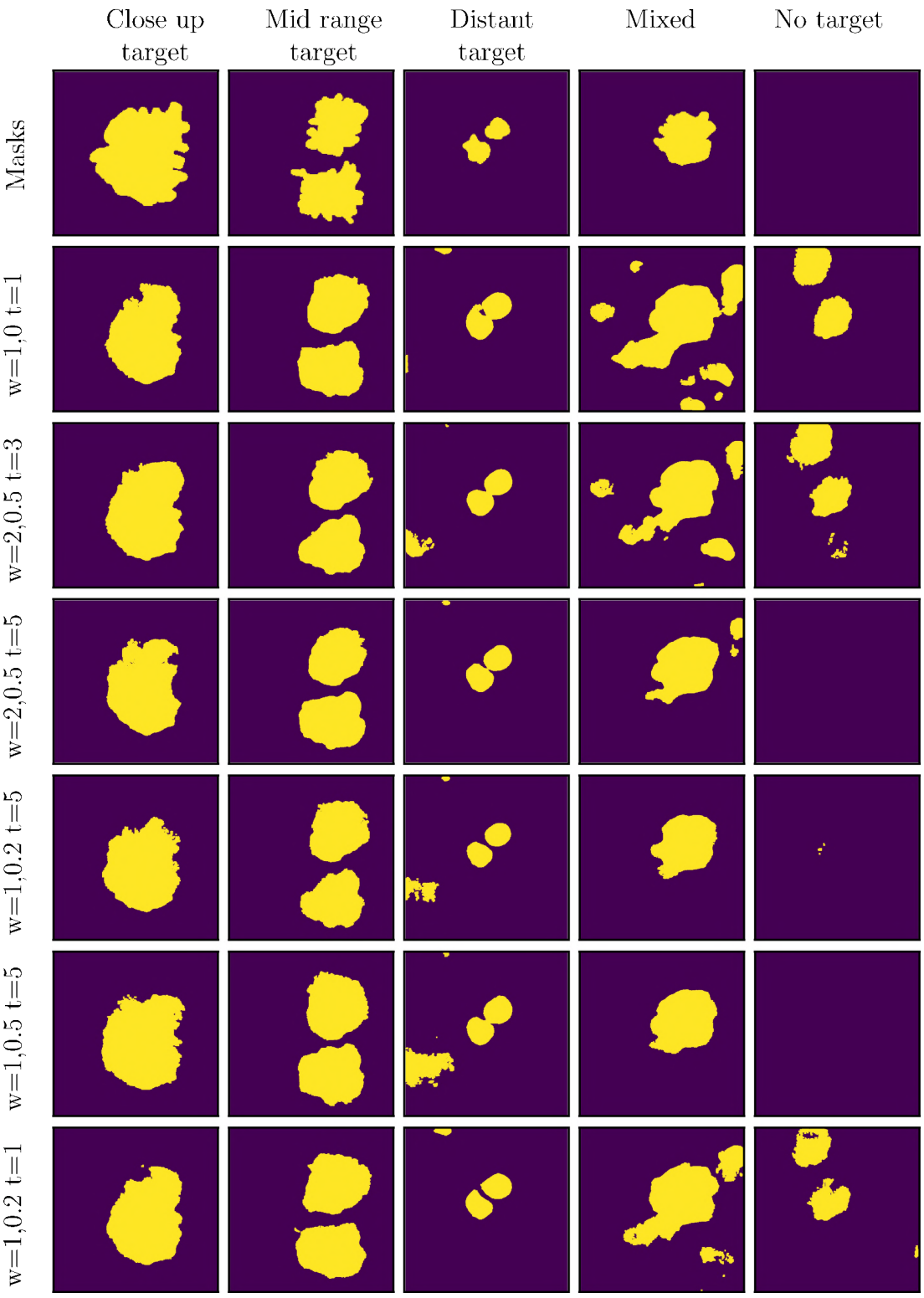


Figure 7.5: Selection of top-performing models using weight map loss.

Table 7.7: Results from k-fold validation for the top-performing models

Batchwise		Precision	Recall	F1	Dice	IOU	Accuracy
WF $\gamma = 1$		0.51±0.16	0.85±0.06	0.63±0.15	0.63±0.15	0.47±0.14	0.92±0.04
t=5	w=2,0.5	0.51±0.11	0.91±0.05	0.64±0.09	0.64±0.09	0.48±0.09	0.93±0.02
Average		Precision	Recall	F1	Dice	IOU	Accuracy
WF $\gamma = 1$		0.44±0.15	0.85±0.06	0.73±0.09	0.47±0.15	0.39±0.13	
t=5	w=2,0.5	0.46±0.10	0.90±0.05	0.77±0.06	0.50±0.11	0.42±0.09	

Table 7.8: Results of assessment of the top-performing models on the unseen test set

Batchwise metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WF $\gamma = 1$		0.88	0.74	0.80	0.80	0.67	0.97
t=5	w=2,0.5	0.81	0.80	0.80	0.80	0.67	0.96
Average metrics		Precision	Recall	F1	Dice	IOU	Accuracy
WF $\gamma = 1$		0.67	0.76	0.79	0.59	0.51	
t=5	w=2,0.5	0.59	0.81	0.83	0.56	0.48	

object-level detections. It is also important that there be as small as possible variance between folds in k-fold validation, as this indicates that the model is stable.

We consider Table 7.7, the results from k-fold validation, and Table 7.8, the inference results. While all average F1 metrics meet the first criterion, none of the batchwise metrics do, and hence, this criterion does not eliminate either model. The second criterion, best possible recall with strong precision, indicates that perhaps w=2,0.5 is the most optimal,

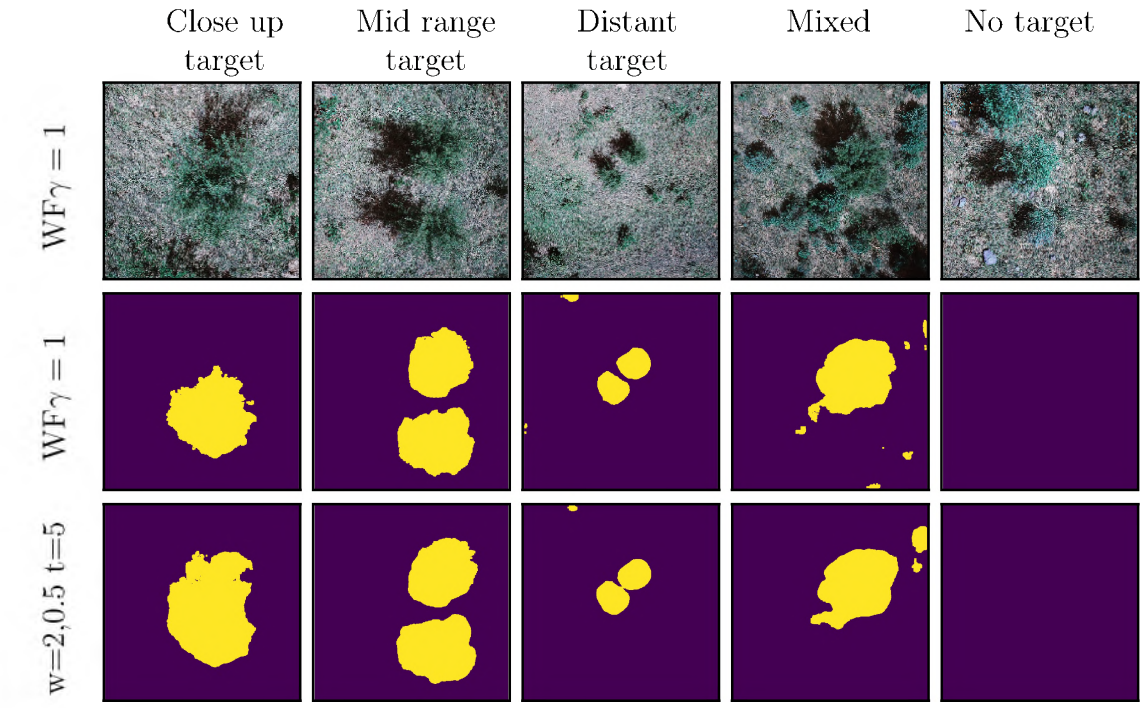


Figure 7.6: Comparison of top-performing models.

as it produces the strongest recall in all categories. Having the best recall means that the top number of pixels belonging to the target class are detected in the image. If we consider Figure 7.6, the higher pixel-wise recall of w=2,0.5 when compared to the WF loss, is apparent in the first column where a fuller detection is made.

When we consider the precision of the models, although model w=2,0.5 beats the WF loss during k-fold validation, it is the other way round on the test set. This is visually apparent by the smaller false positive blobs seen in Figure 7.6, particularly columns two and three, for distant target and mixed images.

As before, there is a trade-off between precision and recall between these two models. Different applications favour different models, and no one model fits all applications.

If a choice must be made, it is more important to have better recall than precision, as false positives can always be filtered out later. As the w=2,0.5 model clearly has the best pixel-wise recall performance with reasonable precision and detects objects more fully than the WF loss, it was selected as the most optimal model for the application of detecting *Hakea*.

7.5 Final model for inference in the field

The final selected training configuration uses the weight map based BCE loss, with hyper-parameters w=2,0.5 t=5. The model for inference in the field made use of the trained weights for this configuration that produced the results in Table 7.6 as a starting point. The model was then trained for a further 150 epochs using dataset 2, as this virtually doubled the number of training samples, thereby hopefully improving the model performance and generalisation ability for field testing.

7.6 Summary

This chapter aimed to determine whether the performance of the baseline model could be improved. Brightness augmentation was found to be beneficial to model performance as it improved the ability of the model to generalise to the unseen test set, which was less bright than the training set. Loss functions that accounted for class imbalance were generally found to improve model performance substantially compared to the use of regular BCE loss, allowing the model to learn greater invariance to fluctuations within the target

class through the down-weighting of noise within the background class. Accounting for uncertainty within the annotation masks at edge pixels by down-weighting the contribution of these pixels using weight map based loss allowed a small additional improvement in performance compared to the regular WBCE loss. The hyper-parameter configuration w=2,0.5 t=5 was selected as the most optimal model through assessment of both quantitative and qualitative results. This model was then trained on all available data in both dataset 1 and dataset 2 for inference in the field.

Chapter 8

Field Testing

In this chapter, testing to determine the best conditions for the operation of the chosen model is reported. Variables pertaining to varying environmental and image-capture conditions were examined to give the best possible indication as to the robustness of the model to these varying factors. The aim of the experiments conducted was to determine which factors caused changes within model performance and in what range of conditions the performance of the model was most favourable. A sample screenshot, showing the application when deployed in the field, is shown in Figure 8.1,

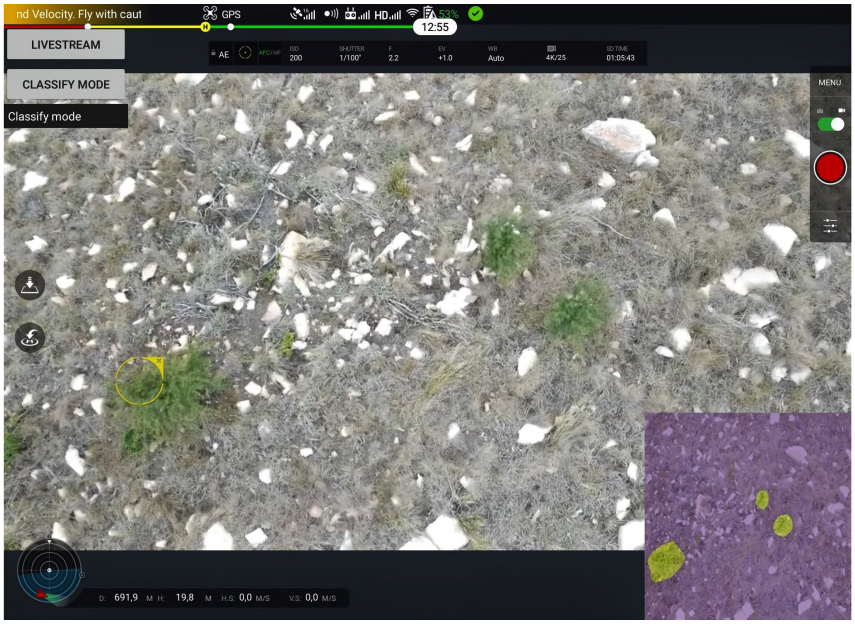


Figure 8.1: A screenshot of the application when deployed in the field in classify mode. The bottom right-hand corner of the first person view shows the segmentation output of the sampled video livestream.

8.1 Experiment 1 - image-capture variables

The exposure set by the cameras and altitude at which the drone was positioned above the target plant was shown in initial testing, as reported in Chapter 6, to have a strong influence on model performance. In this experiment, we conducted a more rigorous experiment to define how the model performance is affected by these factors, altering the altitude above the ground and the exposure value (EV) by set increments.

The exposure value of an image comprises the image ISO, shutter speed and aperture size, each of which controls how much light enters the camera or how sensitive to light the camera is. On DJI drones, the camera can be set to auto-mode, within which it is possible to adjust the exposure compensation value between -3 and 3, with an EV of 0 the recommended setting (DJI Support, 2017). Higher values make the image lighter, while lower values make the image darker.

8.1.1 Methodology

Eight individual target plants were used in this experiment. For each of these, four altitudes above ground were tested, 8 m, 13 m, 18 m and 23 m, and at each of these altitudes, a range of five exposure values were tested, in the EV range -2 to 2. The target plants were divided equally into two time slots, one early in the morning and one near mid-day, so as to capture the two extremes of naturally fluctuating luminance within the results. It is noted that particular shrubs were not revisited, instead randomly selecting a new set of four shrubs in each time slot. An overcast day was chosen to eliminate the possible effect of changing shadow on the results.

8.1.2 Results and discussion

During the first time slot, the time of ambient light after dawn before direct sunlight was present, the light intensity increased from 30 to 1 K lux over the time span of approximately 45 minutes. This is apparent in the rightmost column of the images in this time slot (Figures 8.2 – 8.5, where once again yellow indicates predicted membership of the target class, purple indicates a prediction of the negative class and ground truth instances of the target plant are outlined as a visual aid in white), as it is seen that the brightness of the images increases for each consecutive target even though EV is the same. The second time slot in the late morning (Figures 8.6 - 8.9) had more consistent light intensity, with

fluctuations between 16 K and 18 K lux, which is reflected by a much more consistent brightness to the eye in the rightmost column (brightness is perceived by the eye on a logarithmic scale). It is interesting to note that the number of positive detections at 8 m increases with the light intensity, as apparent from the increase in the yellow colouring of predicted positives when observing targets 1 to 4. Targets 1 and 2 do not reach the same apparent level of exposure as the other targets, despite the same camera settings, as the low light intensity present at the time of inference limits the maximum level of light entering the aperture.

Considering all targets, apart from the first two in the early morning slot, there appears to be a trade-off between precision and recall as exposure increases from -1, with predictions increasing in precision with increasing exposure, but decreasing in recall. A particularly clear example of this is seen in the case of target 5, Figure 8.6. At all altitudes for this target, but particularly clear at 18 m, it is seen that at EV -1 there are extensive positive predictions which, while covering true positive regions well, do so at the cost of a substantial region of false positive prediction. Thereafter, as the EV increases, the false positive regions reduce, which improves the precision score, until finally at EV 2 no false positives are detected in Figure 8.6, but at the cost of missing some true positives as well. In this instance, it is clear that at EV 1 there is a good balance between precision and recall, with all targets detected with very few false positives.

The extreme values of exposure, EV -2 and 2, produce poor predictions. Very few true positive predictions are made at EV -2, likely due to the necessary features being indistinguishable from other features, resulting in both few and poor predictions. In contrast, at EV 2, while some target plants are detected well at EV 2 – especially with targets 7 and 8, many target plants are missed.

EV 0 and 1 both offer strong performance and for the most part, the same plants are detected as positive, but with different trade-offs, as EV 0 offers slightly higher detections of both true and false positives, while EV 1 offers the opposite. However, in many cases the slightly stronger precision predicted when using EV 1 results in better predictions, as the true positive detections are maintained but are tighter at higher exposure, resulting in fewer false positives. Thus, depending on the application, exposure values between 0 and 1 can be used to achieve the best performance.

For all of the targets except the first two, the target is detected in the 8 m image, at least at one exposure. If we limit this to either exposure 0 or 1 at this altitude, then the best detection across all altitudes is made at exposure 0 for targets 3, 4 and 5 and an equally acceptable result is provided at both EV 0 and 1 for targets 6, 7 and 8, with slightly

higher precision at EV 1 as expected. Furthermore, at 8 m, the exposure choice appears to be the difference between a true positive detection or a false negative detection, in addition to an effect on the false positive rate.

Considering EV 0 and EV 1 at 13 m, targets 1, 2, 6, 7 and 8 are best detected at EV 1, while target 5 is best detected at EV 0 and there is nothing to promote one exposure over the other for targets 3 and 4 at this altitude. The difference between the two exposures affects true positive detection to a smaller extent at 13 m than at 8 m but has a definite effect on the trade-off between false positives and false negatives.

At 18 m all predictions, except for those in Figure 8.3, were better detected at EV 1 and generally with fewer false positives than at EV 0. At all except the first two targets, the difference between the exposure values again appears to affect the tightness of predictions along with the false positive rate.

At 23 m compared to 18 m, performance is lost through a decrease in the true positive detection rate and EV 1 produces better predictions at all except targets 2, 3 and 4. At the first of these, neither detection is very good, while the latter two produce better predictions at EV 0. Overall, it appears that EV 0 is better for low altitudes of 8 m, while EV 1 is better for the majority of targets at higher altitudes. Although higher altitude appears to give a better result generally, as in the case of target 5, this is not always the case, as with target 2. However, it is clear that if a survey were conducted at an altitude between 13 m and 18 m with an exposure value between 0 and 1, a high true positive detection rate could be obtained, with false positives minimised at EV 1.

Unfortunately, a high number of false positives are detected by the model, indicating that while the model is good at detecting the target plant, further measures are necessary to remove false positives. Such methods could include the use of three classes when annotating, with other shrubs as an additional class to the existing target shrub and other classes. This may assist the loss function during training with greater importance given to distinguishing between the target and other shrubs, rather than treating other shrubs in the same manner as any other background vegetation. Further inclusion of true negative shrub samples in the training set may also assist the model in differentiating between the shrubs. Another method for eliminating false positives would be to apply a classifier, such as those trained in Chapter 5 to the segmented image, with each positive detection fed to the classifier.

8.2 Experiment 2 - environmental variables

Natural light intensity and shadow length are both factors which may result in performance variations. To investigate these effects, the model was tested at three different times of the day, namely, 8 AM, 12 PM and 3 PM. These times of the day reflect a progression in both light intensity and shadow length, with 8 AM roughly an hour after sunrise providing a time of low light intensity and long shadows, 12 PM a time of maximum light intensity and the shortest shadows, and 3 PM a time of both in-between light intensity and shadow length. Thus, we consider the time slots in the order: 8 AM, 3 PM, 12 PM, such that the sequence reflects increasing light intensity values and decreasing shadow length as the sun moves from a low to high angle.

A number of repeat images were made to further explore this dependency by rotating the drone above each target. Although the individual target plant selected was the same, its orientation within the image was adjusted on each repeat, which allows further insight into how the angle of the shadow may affect the performance. Additionally, this gives an indication of how successful data augmentation strategies were.

8.2.1 Methodology

The drone was flown such that each target shrub fell close to the centre of the image. The exposure value and altitude were kept within the range EV 0-1 and altitude 13 m-18 m, as these were found to be optimal in Experiment 1. For each target plant, the drone was flown to the target and positioned such that the plant's shadow was directly behind it. Inference was performed in this configuration. The drone was then turned such that the target plant's shadow shifted by approximately 90 degrees before inference was performed again. This was repeated twice more, such that four images were taken for each plant. This procedure was repeated for four distinct target plants, which were revisited in each time slot.

8.2.2 Results and discussion

There are a number of variables in play within this experiment, as the time of day is related to light intensity, the orientation of the shadow with relation to the shrub within the image, shadow length and angle of the sun. It is observed from Figures 8.10 - 8.13 that in the images taken in the 8 AM slot, the low light intensity resulted in softer shadows

with less contrast to surrounding vegetation colours than those taken later in the day when the light intensity was higher, while the shadows were longer at 8 AM due to the angle of the sun. As the light intensity increased, the shadows reduced in length, darkened and changed orientation relative to the shrub, indicating that these variables are linked. This makes it difficult to infer that a change in performance is directly due to one of the variables, rather than due to the combination of several.

As a result of these factors, the appearance of the target shrub itself differs between the time slots, owing to both the light intensity and angle of the sun, illuminating different features and bringing shadows between branches within the shrubs themselves into relief. Furthermore, the target shrubs used for inference grow on a slope and as such, even though the drone's camera faces directly downwards, changes in the image beyond shadow orientation occur between the four tested orientations of a single shrub, within a single time slot. This results in a slight change in the prominence of features, and it is observed that there are distinct variations in performance between them, an example of which is easily seen in the 8 AM and 12 PM slots in Figure 8.10.

A contraction of shadow length from the 8 AM to 12 PM slots is apparent, as expected with the sun near its zenith. However, in the midday time slot, a portion of the shadow is still visible as the sun does not pass directly overhead in winter. This shadow would be present to a lesser extent in the summer months, with the sun nearer to directly overhead.

Considering each shadow orientation across all three time slots, target 9 (Figure 8.10) illustrates that the 8 AM and 12 PM slots are better than 3 PM for detection. The best detections across all positions, for this target, were made at 12 PM. The poor performance at 3 PM indicates a long dark shadow is worse for performance than a long light shadow, whereas performance at 12 PM compared to 8 AM suggests that a short, dark shadow is better for performance than a long light one. However, in experiment 1, when flights were made both in the early morning and at midday on an overcast day, a far higher number of positive detections overall were made than in experiment 2. The shadows in experiment 1 were extremely pale due to the overcast sky, which suggests that very pale shadows do not lead to poor predictions.

Target 10 illustrated relatively good detections of the target at all time slots, along with several false positive detections. In this case, the shadow at 3 PM was broken by another shrub, which may have prevented the poor performance that was observed for target 9 at this time slot. Target 11 was poorly detected at all time slots, possibly due to the presence of large shadows generated by a nearby tree, but may also have been due to the

quality of the light, which is quite different from that in the 3 PM slot for targets 9 and 10. A single correct detection was made at both 8 AM and 12 PM at position 2. The model failed to detect target 12 in any time slot, indicating that the model has room for further training. It is noted that the shadow at 12 PM for this target is still long due to the angle of the slope, which may have negatively affected its performance.

Overall, the observations in this experiment indicate that long dark shadows produce the worst performance. Long pale shadows are not as harmful to performance, with some good predictions made. The best predictions in experiment 2 were made near midday when the shadow is the shortest, which corresponds to the training set upon which the model was trained. However, the 8 AM slot yielded reasonable results as well, indicating that pale shadows can be accommodated by the model. The very pale shadows in experiment 1, when the sky was overcast, indicate that the less observable a shadow is, the better it is for the quality of the prediction.

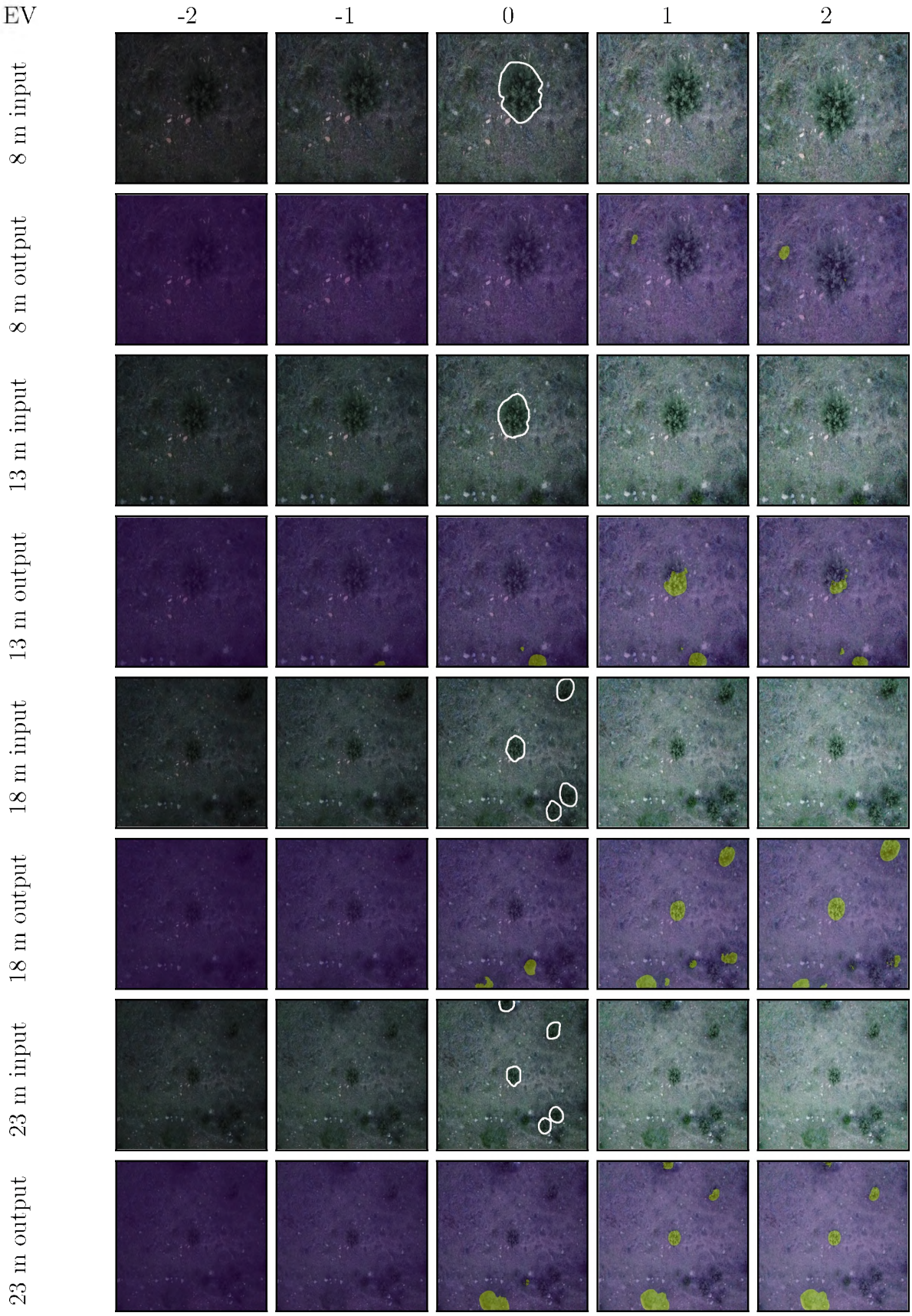


Figure 8.2: Target 1 (Exp 1) - inference performed during the early morning session.

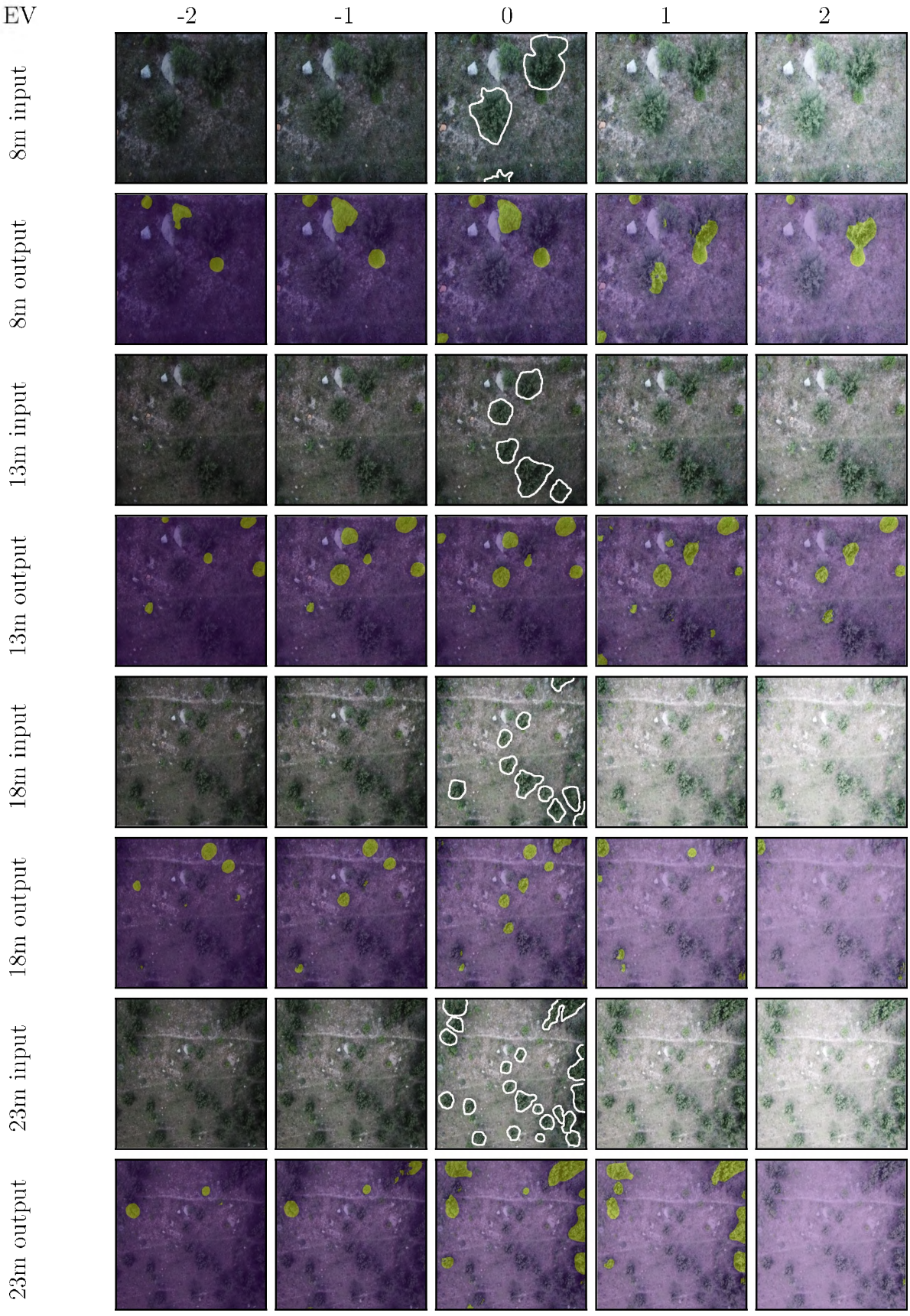


Figure 8.3: Target 2 (Exp 1) - inference performed during the early morning session.

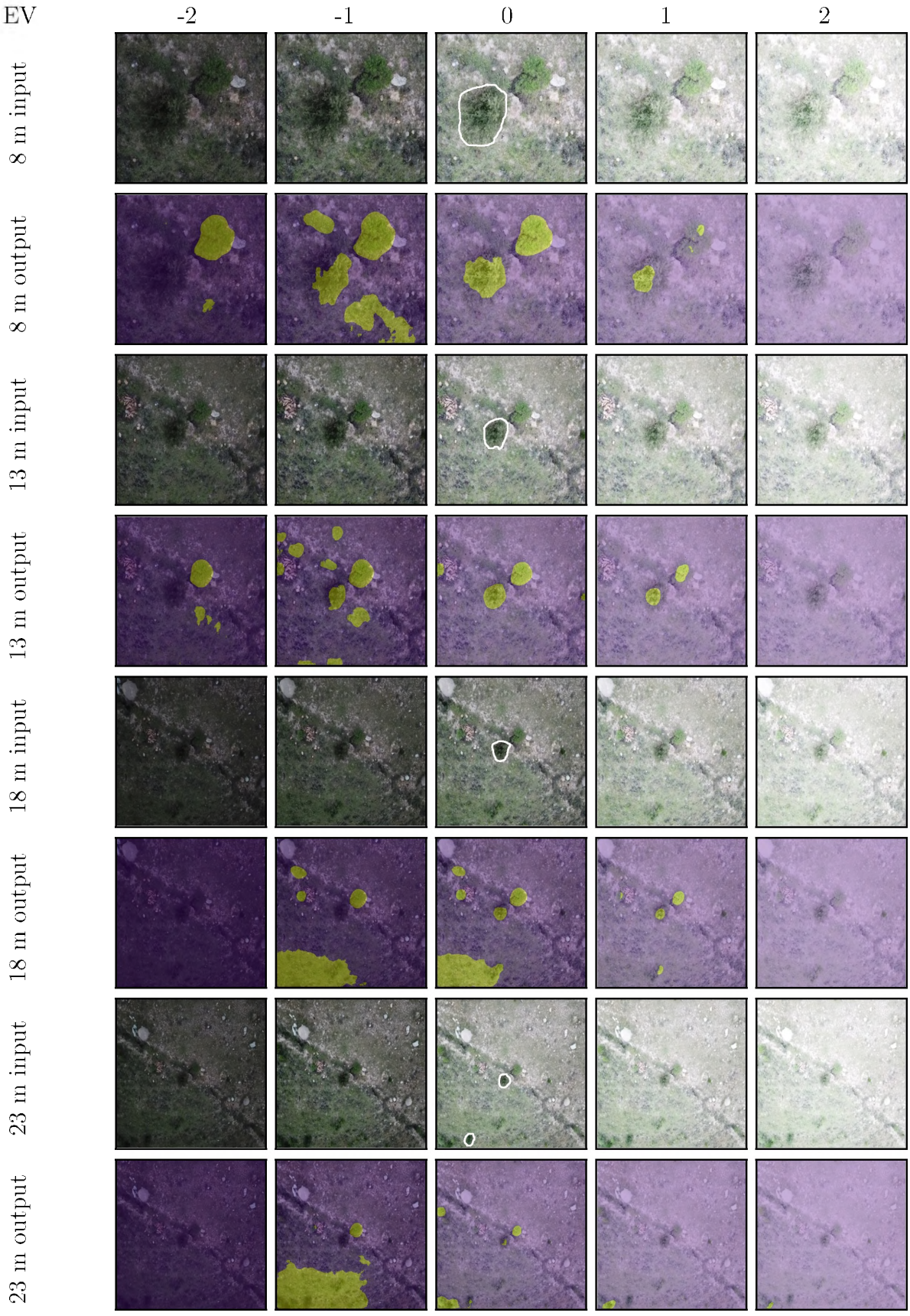


Figure 8.4: Target 3 (Exp 1) - inference performed during the early morning session.

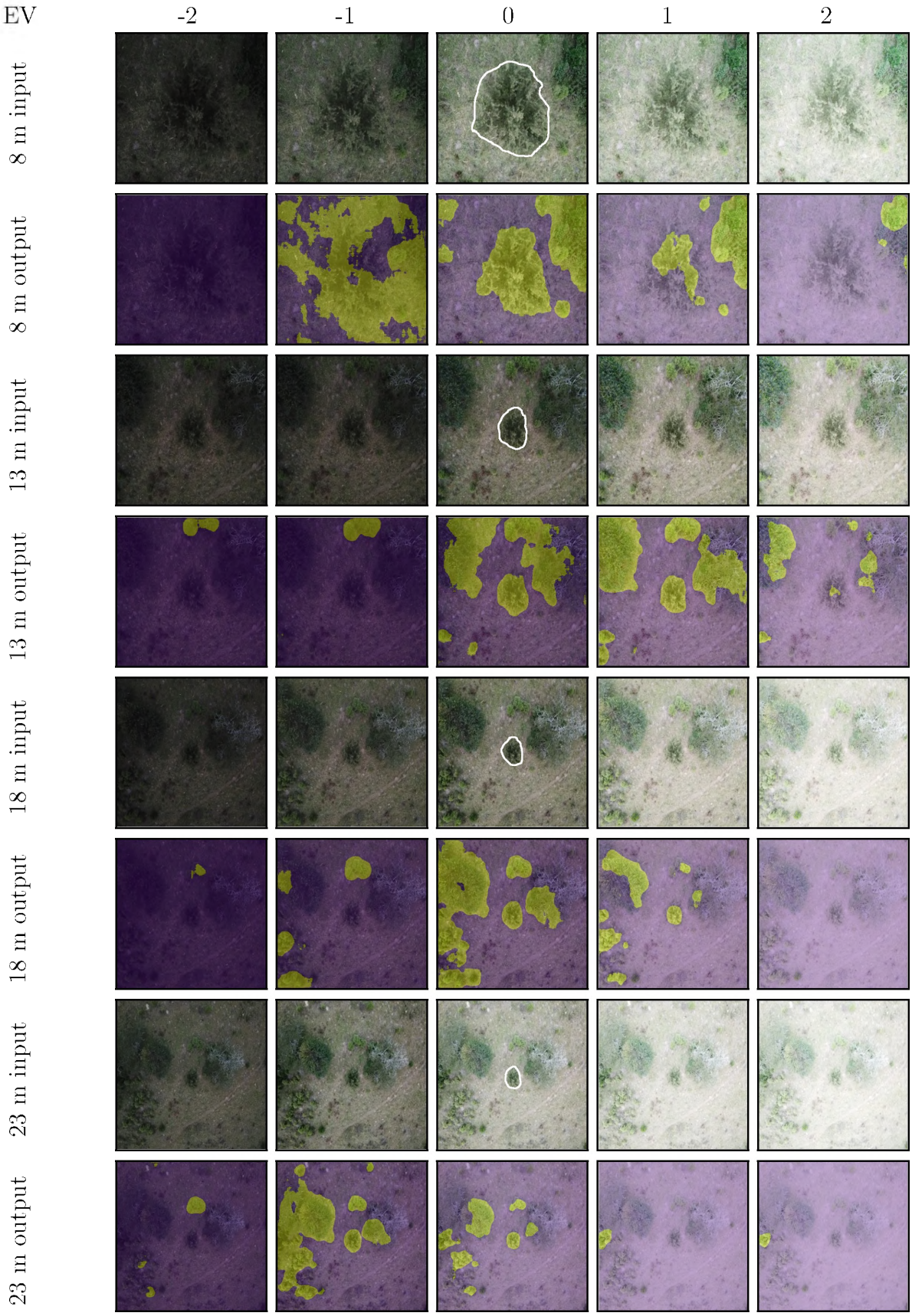


Figure 8.5: Target 4 (Exp 1) - inference performed during the early morning session.

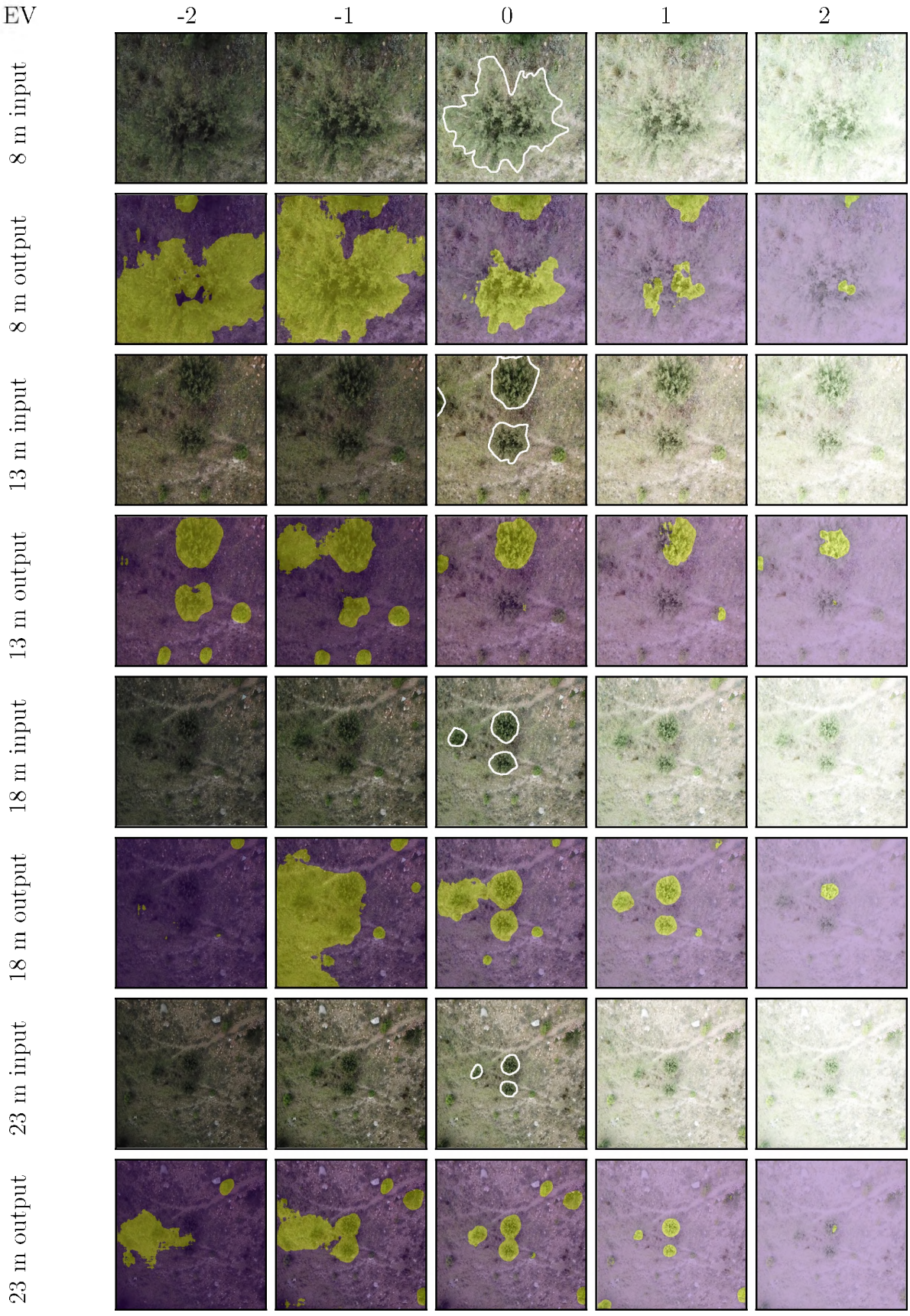


Figure 8.6: Target 5 (Exp 1) - inference performed during the late morning session.

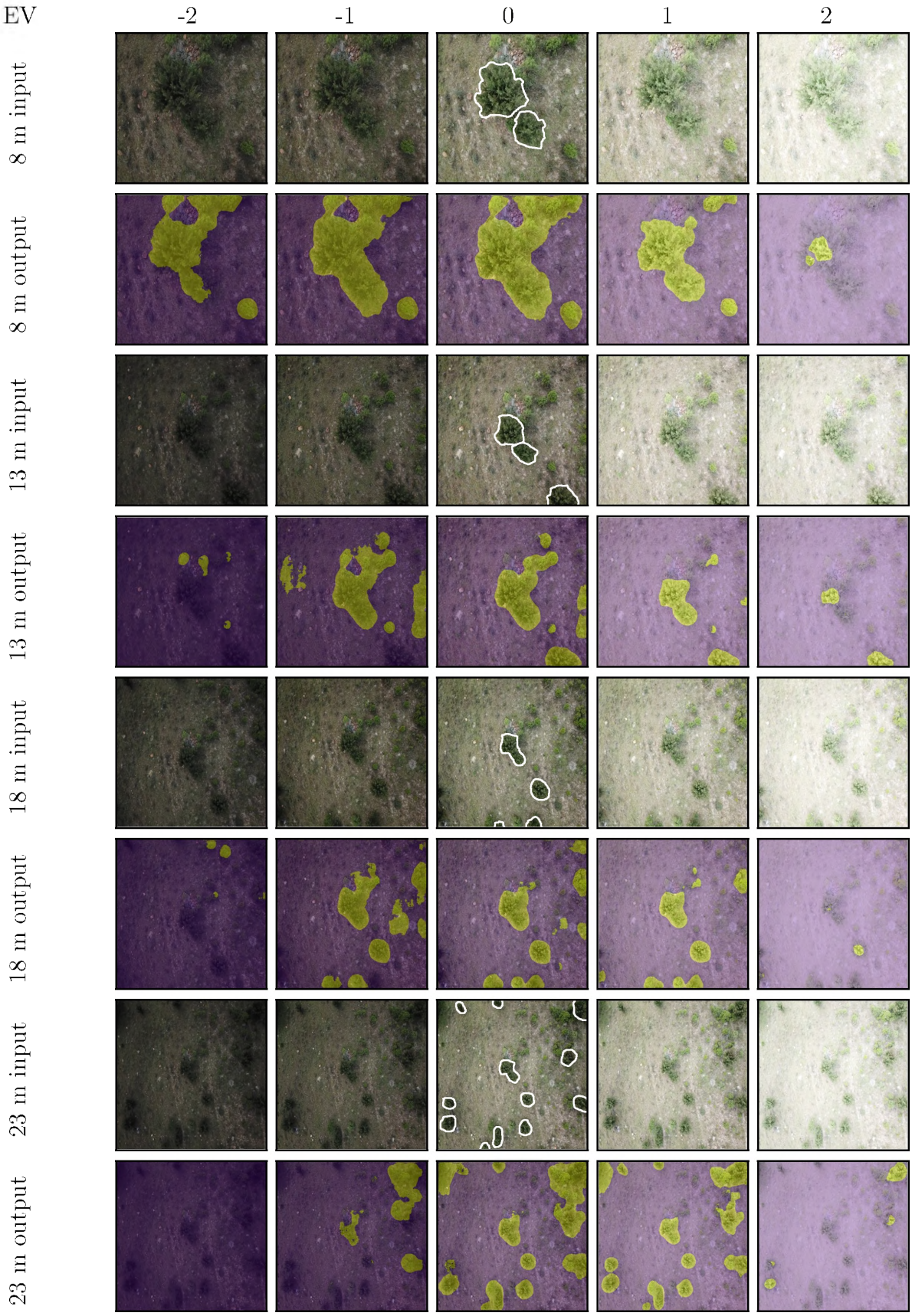


Figure 8.7: Target 6 (Exp 1) - inference performed during the late morning session.

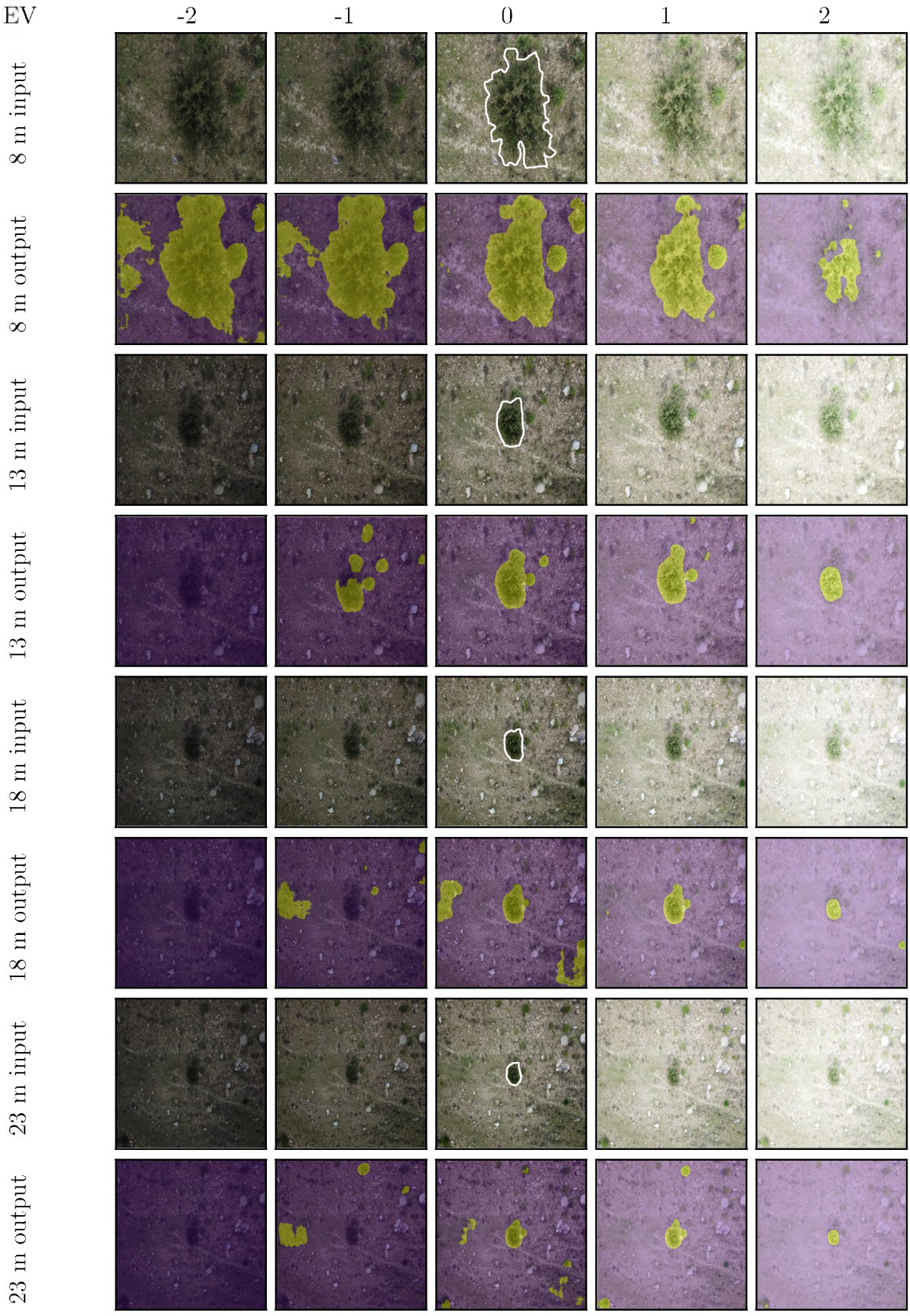


Figure 8.8: Target 7 (Exp 1) - inference performed during the late morning session.

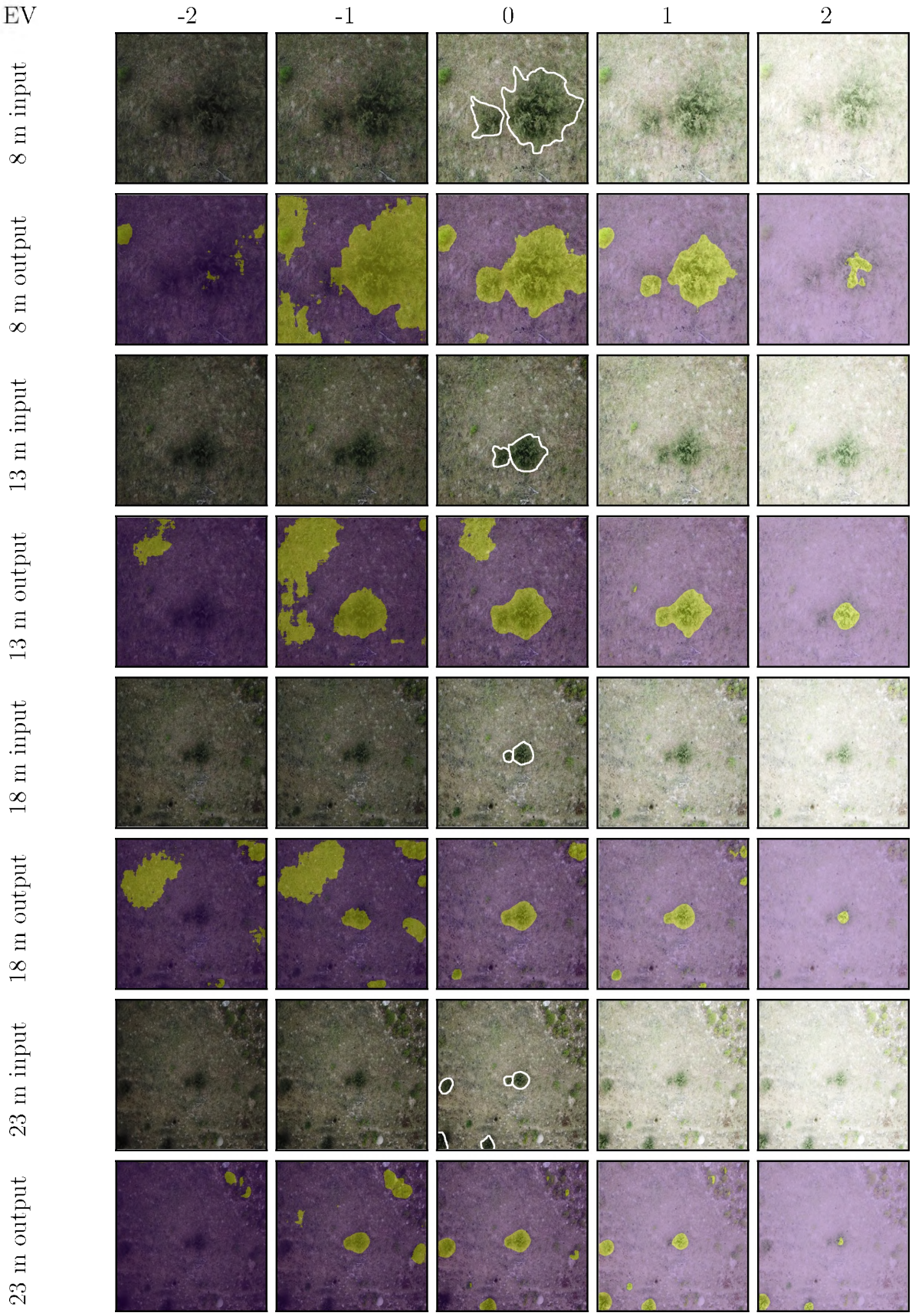


Figure 8.9: Target 8 (Exp 1) - inference performed during the late morning session.

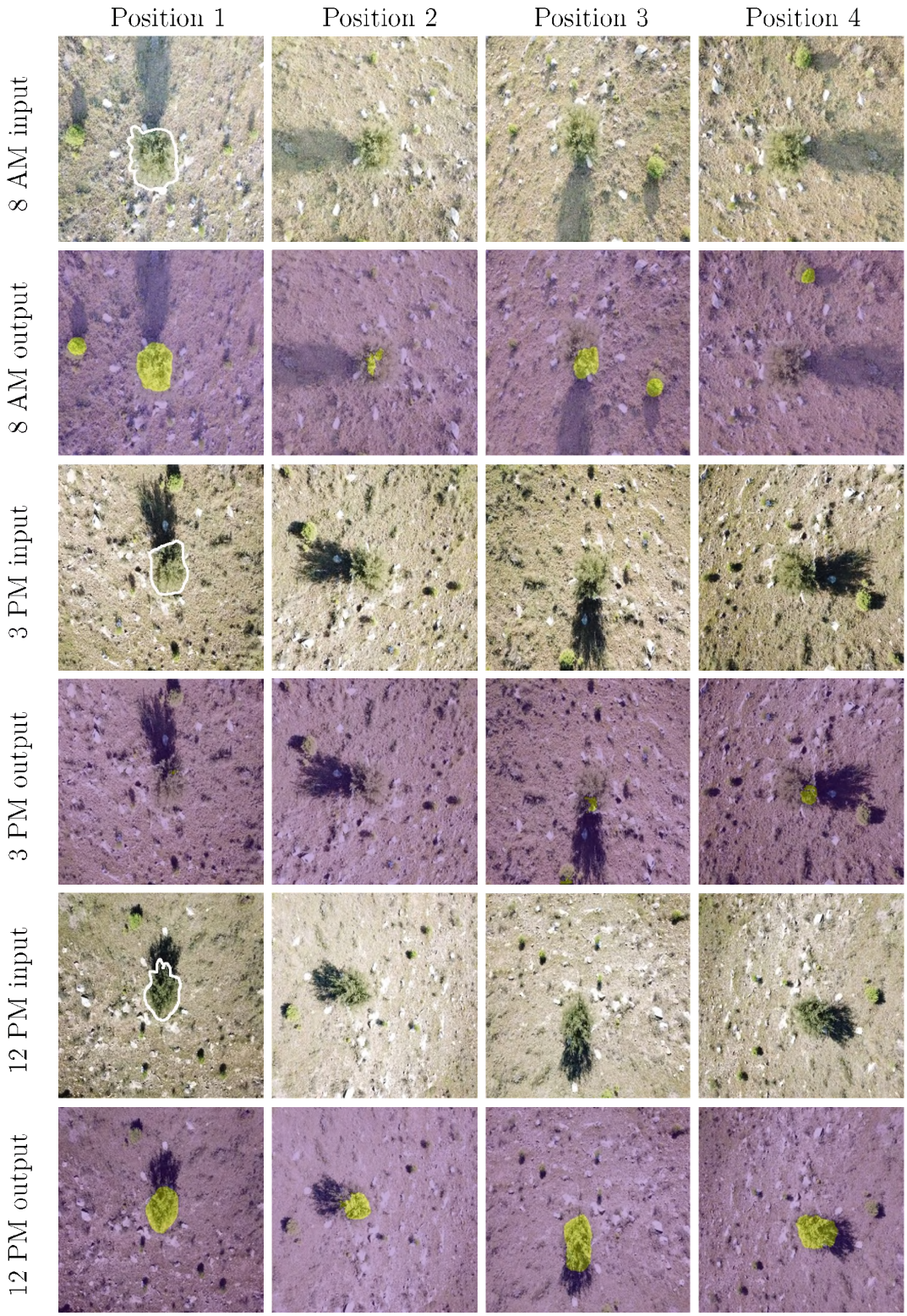


Figure 8.10: Target 9 (Exp 2) - inference performed on a sunny day where shadows are evident.

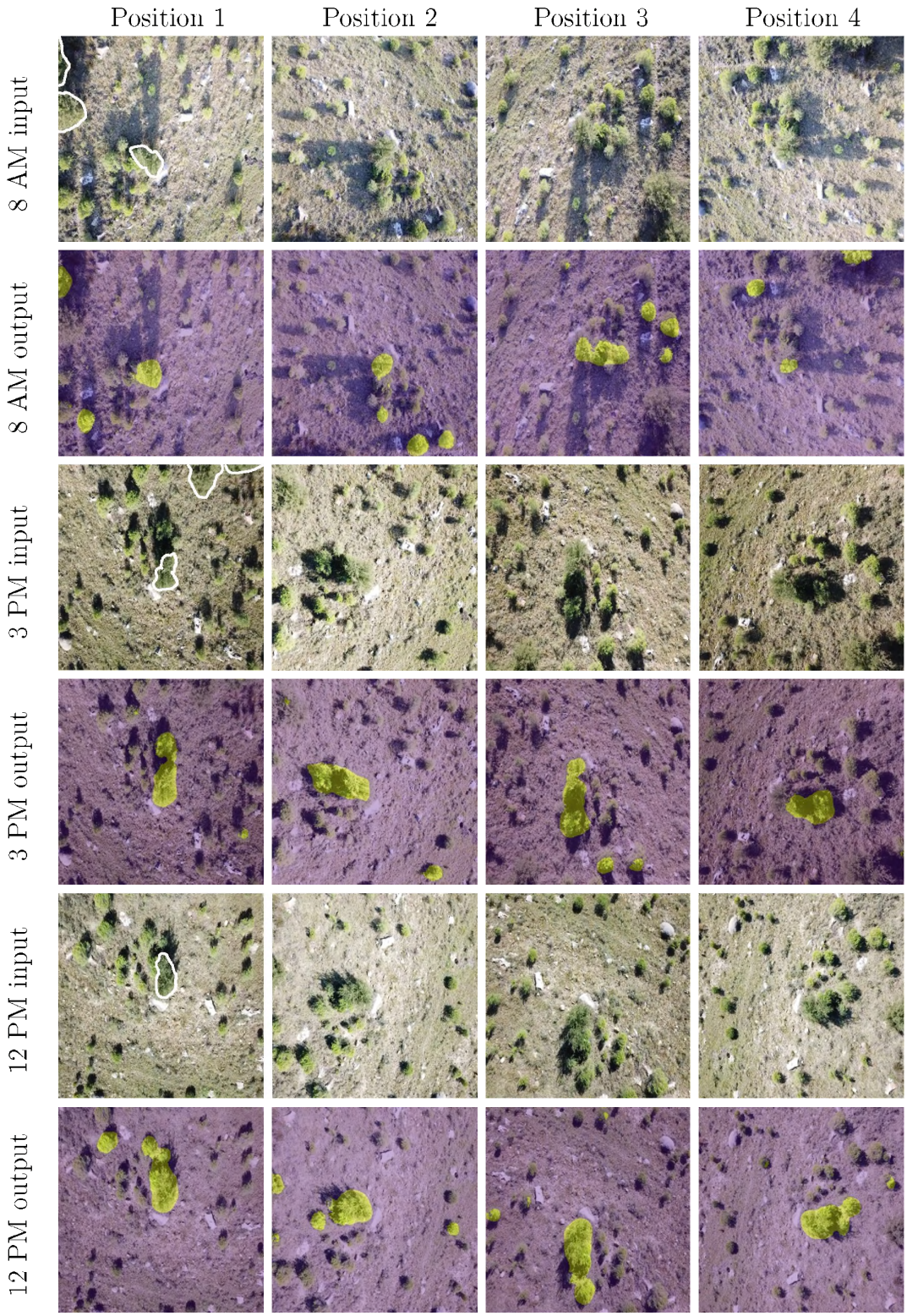


Figure 8.11: Target 10 (Exp 2) - inference performed on a sunny day where shadows are evident.

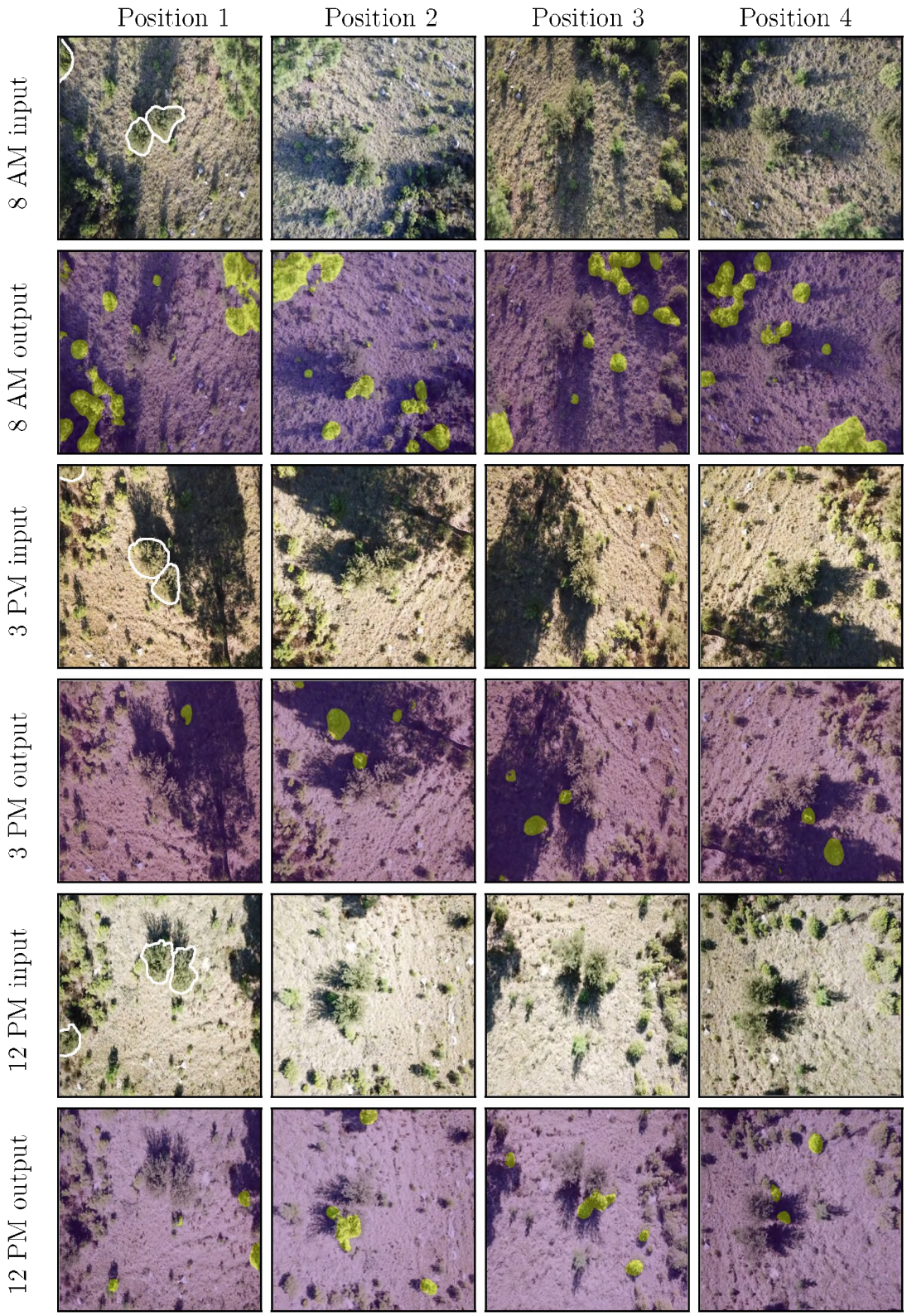


Figure 8.12: Target 11 (Exp 2) - inference performed on a sunny day where shadows are evident.

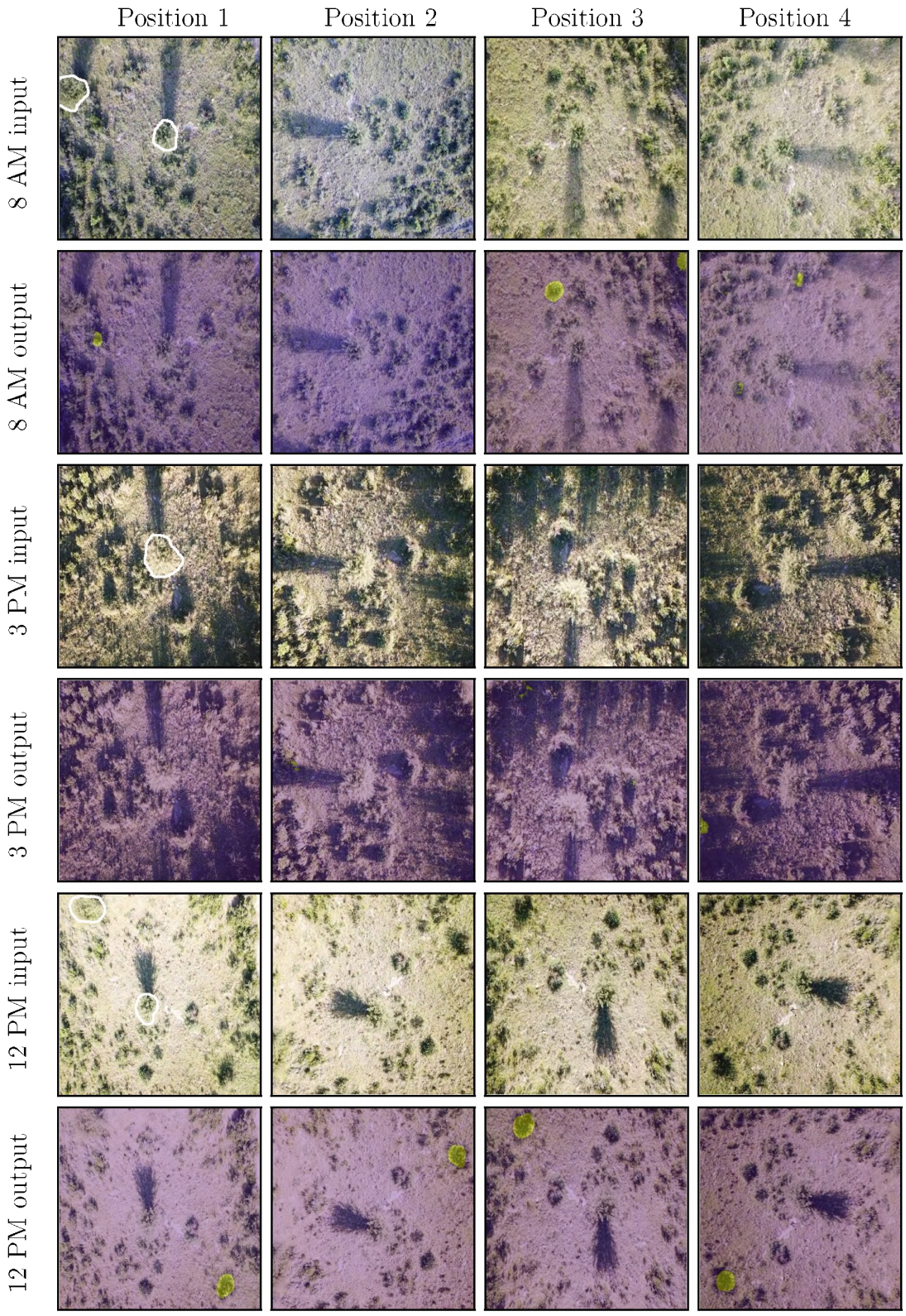


Figure 8.13: Target 12 (Exp 2) - inference performed on a sunny day where shadows are evident.

8.3 Optimal operating conditions

Based on the two experiments performed, the best operating conditions for the model were deduced to be:

- Flight at an altitude between 13 m and 18 m.
- Camera exposure value of 0 or 1.
- Flight when shadows are short (near midday) or very pale (early morning or on an overcast day).

To obtain a quantitative representation of model performance, we consider the subset of images in both experiments that meet these operating conditions. Cases from experiment 1 were extracted at 18 m with EV 0 and from experiment 2 at 12 PM, the first column. The ground truth and predicted positives were counted and recorded in Table 8.1. This gives a reflection of model performance under real conditions within the determined flight parameters. All targets marked with an asterisk were from the early morning slot, while the remainder were from the late morning (12 PM) slot.

From this table, it is seen that 18 of 33 targets were detected, yielding 54.5% overall true positive detection, in real field conditions. However, the majority of images had a 100% true positive detection rate, indicating overall success, although further development of the model, particularly to obtain greater invariance to rotations, would be beneficial.

Table 8.1: A quantitative representation of model performance in the field under the recommended operating conditions, sampled from experiments 1 and 2

Target	**Ground truth**	**True positive detection**	**Recall rate** (%)
1	4	0	0
2	10	4	40
3	1	1	100
4	1	1	100
5*	3	3	100
6*	4	4	100
7*	1	1	100
8*	2	2	100
9	1	1	100
10	1	1	100
11	3	0	0
12	2	0	0
TOTAL	33	18	54.5

8.4 Contextualisation of research achievements, limitations and challenges

Overall, the U-Net model in its selected configuration with weight map-based loss proved successful in its task of detecting the target shrub *Hakea*. During k-fold validation on the dataset, a recall rate of over 90% was achieved for both batchwise and average calculations of the metric, while on the test set over 80% was achieved. When the model was tested in the field within the constraints found to give the best performance, an overall recall rate of 54.5% was achieved. However, a 100% recall rate was achieved under real conditions, within the set of those recommended, on 8 of the 12 images.

Furthermore, an accuracy of 0.93 ± 0.02 was achieved on k-fold validation and 0.96 on the test set, while an F1-score of 0.64 ± 0.09 (batchwise) and 0.77 ± 0.06 (average) was achieved for k-fold and 0.80 (batchwise) and 0.83 (average) F1-score on the test set.

8.4.1 Challenges and limitations

Numerous challenges were encountered during the process of this research, particularly with regard to detection in the field. When operating the system in a real outdoor environment, factors such as season and cloud cover have impacts on model performance. Season changes affect multiple variables, such as the dryness of the environment (which in turn affects the colour of the vegetation in the area), the angle of the sun at midday (having an effect on shadow length), the light intensity and phenological traits such as flowering. Cloud cover and illumination intensity affect the brightness of the landscape and the intensity of shadows, particularly of taller vegetation such as shrubs. While some of these challenges, such as variations in brightness, could be addressed during the training process, others cannot unless the dataset reflects these variations sufficiently.

As the dataset used in this thesis was collected over two days in the summer months, this limited the ability of the model to be robust to seasonal variation. However, field testing was conducted in early winter, and most of those images assessed in Table 8.1 achieved perfect recall rates, which bodes well for its performance in summer months. Unfortunately, a high number of false positives were detected, which indicates that further model refinement is necessary. It is possible, but untested, that if tested in the field in summer months when conditions are more similar to those in the dataset, that the false positive detections would be reduced and the recall rate increased. The target plant flowers

in late winter, a season not reflected in the dataset, and as such the model developed is not suited for use during the flowering period.

Furthermore, a plethora of different models, loss functions and hyper-parameter tunings exist, an exhaustive evaluation of which is impossible in the time-frame available for this research. In this thesis, we have attempted to rationally select only those most relevant to investigate but acknowledge that there may exist other models or approaches that have not been considered.

Despite these limitations and challenges, a deep learning-based model was successfully trained and integrated into the drone system, which was capable of detecting the target shrub in the field while the drone was in flight, demonstrating the viability of the system.

8.4.2 Comparison to key literature

To the best of our knowledge, there has been no previous use of intelligent drones for in-field detection of invasive plants, although there has been substantial research within the remote sensing community in detecting alien vegetation in aerial imagery, as outlined in Section 3.2.3, often to produce distribution maps from orthomosaics. The approach taken in our research of classifying or segmenting a single image rather than an orthomosaic is not a common practice in literature pertaining to the detection of invasive plants, with the only instance of this found performed by Baron *et al.* (2018) in their research on the detection of *Iris pseudacorus*. This study calculated a set of 68 features per pixel from RGB imagery, which were classified in a PBIA approach using a random forest classifier. This contrasts to our research in which a deep learning-based semantic segmentation approach was used, which learnt in a supervised fashion which features to extract. The study by Baron *et al.* (2018) achieved an overall accuracy of 99%, precision of 4.8% and recall of 93%, whereas in our study an accuracy of 96% was achieved on the test set, with a batchwise precision of 81% and recall of 80% and average precision and recall of 59% and 81% respectively. Although the recall and accuracy achieved in our research are lower, our precision value is considerably higher. In both pieces of research, however, precision is the metric which suffers compared to the others, indicating a likelihood for the detection of false positives.

Research on mapping *Hakea sericea*, one of two of the species of *Hakea* prevalent in our study area, as one of seven land cover classes was conducted by Alvarez-Taboada *et al.* (2017), in which precision and recall above 75% were achieved on the orthophoto created

from drone sourced imagery. This result is similar to the over 80% precision and recall achieved by our model on the test set. Alvarez-Taboada *et al.* (2017) noted that woodland was often misclassified as *Hakea*, which indicates that false positives were a shortcoming of their approach, as was the case in our research as well. In their study, a set of chosen features were extracted from objects, some of which drew on the available NIR band, which was not available in our study.

Low sun elevation angle has previously been noted as unfavourable for data collection by Lehmann *et al.* (2017) and Chabot *et al.* (2018) who recommended that flights be conducted within two hours of the sun's zenith to achieve the best possible illumination. This recommendation agrees with the findings of our study, in which we found that long dark shadows result in poor model predictions. However, as extremely low sun elevation angle creates pale shadows in the case of our target plant, we include the caveat that low angles are only detrimental to performance if the light intensity is sufficiently high to cause dark shadows.

Chapter 9

Conclusion

With the ever-growing threat of invasive alien plants to natural ecosystems and water reserves, monitoring and management of these species are growing in importance. Current detection strategies using remote sensing techniques tend to be based on older machine learning techniques, rather than utilising the power of deep learning which has shown favourable results compared to older techniques in other areas of application. These methods also generally require powerful processing to obtain orthomosaics, limiting their usability to the creation of maps and associated calculations. The use of drones for populations counts, as a remote sensing tool and in precision agriculture suggests potential for their greater integration into the monitoring and management of invasive species, beyond acting as a source of high-resolution aerial imagery. This suggests room for the development of intelligent drones capable of in-field detection of invasive plants, which would facilitate such applications.

9.1 Summary of research

The intention of this research was to develop and test such a system, determining to what extent a reliable detection of a particular target genus of invasive plant could be made using an intelligent drone, as well as what the best way to approach this problem would be. We evaluated two approaches, namely classification and segmentation, as ways of detecting the target. The classification approach was promising, but not viable in practice as the algorithms tested for candidate proposal were unsatisfactory. The segmentation approach was better in this regard, as it did not rely on these algorithms. The segmentation model was then integrated with the DJI Mobile SDK to produce an Android application, which

allowed detections to be made in the field.

Based on findings from a preliminary field test, the segmentation model was refined to improve its performance by assessing various loss functions to account for class imbalance and uncertainty in edge annotation. The final model was then evaluated in the field to determine the system's suitability to the task of detecting invasive vegetation.

9.2 Achievement of objectives

The principal objective of this research was to determine whether a commercially available drone augmented with a deep learning model was able to detect invasive plants in the field. This was approached through three sub-objectives, the attainment of each of which is detailed below.

1. **Determine which deep learning approach is best suited to invasive vegetation detection** – Four classification models, known for their strong performance, were initially assessed on ideal candidate windows containing only instances of the target and other shrubs, and no other vegetation or land cover types. This allowed for an observation to be made on whether these two classes were indeed separable or not. A good detection was made using MobileNet, with a high accuracy and F1-score of 0.87 ± 0.07 and 0.85 ± 0.10 respectively achieved from k-fold validation, as well as respective scores of 0.98 and 0.97 on the unseen test set. This indicated that the MobileNet model is capable of successfully differentiating between the target and other shrubs. The classification approach, however, relies on other algorithms to propose candidate windows. Object proposal algorithms, threshold-based algorithms, and vegetation indices were trialled as possible algorithms to perform this function, but none was found to provide windows that proposed only shrubs. A segmentation approach was then examined, using the U-Net architecture, which proved more suited to the task due to pixel-wise classification.

2. **Investigate what optimal level of performance can be achieved in detection** – Different loss functions were evaluated to obtain the best possible performance using the U-Net deep learning-based segmentation network. The strongest performing model, both in terms of quantitative and qualitative results, was found to use a configuration with weight map-based BCE loss and associated hyperparameters w=2,0.5 t=5. This model was able to achieve an accuracy of 0.93 ± 0.02 and F1-score of 0.64 ± 0.09 (batchwise) and 0.77 ± 0.06 (average) from k-fold valida-

tion, while on the test set it achieved 0.96 accuracy and 0.80 (batchwise) and 0.83 (average).

3. **To investigate the suitability of the augmented drone system for detection in the field** – The augmented drone system, consisting of the trained model and DJI Mavic Pro integrated via an Android-based application, was trialled in the field. It was found that a prediction could be produced in approximately 5 s when a frame was sampled from the livestream as input to the model. From the examination of various environmental and image-capture conditions, a set of optimal operating conditions were determined, and within these operating parameters, the system was found capable of good performance on the whole and thus suitable for the task, although false positive detection remains an issue.

The achievement of the sub-objectives indicates that the primary objective has been achieved, namely, that a commercially available drone augmented with a deep learning model is able to detect invasive plants in the field.

9.3 Contributions of research

In this thesis, we have shown the capacity for drones to detect invasive alien plants in the field, through the augmentation of a DJI Mavic Pro with a deep learning-based segmentation model and applied in the case of the invasive shrub *Hakea*. A dataset was created from aerial images of vegetation, containing both the target shrubs and other shrubs, amidst other vegetation, to facilitate the supervised training of models. Through the evaluation of different CNN architectures on this dataset, it was demonstrated that CNN based classifiers could distinguish between the target shrub and other shrubs when shrubs are contained in ideal candidate windows. Moreover, MobileNet was shown to be suitable for this task. Furthermore, the suitability of deep learning-based segmentation was demonstrated for vegetation detection. Loss functions that account for class imbalance were shown to be important, and a novel loss function was introduced to account for uncertainty in edge annotation, based on pre-computed weight maps. Sections 7.2.2 and 7.4.3, in which this loss function was introduced, have been published in a modified form in (James & Bradshaw, 2019). Finally, the optimal conditions for the use of the system in the field were also established.

9.4 Future work

Future work could build upon that reported on in this thesis to develop a model that is robust to all seasonal changes, including flowering. Furthermore, a transfer learning approach could be investigated as a means of developing a system to detect other invasive, or indigenous, plants. Further investigation into the development of a loss function or experimentation with a third class to reduce false positives would be valuable. Development of suitable candidate window proposal algorithms would allow the alternative classifier approach to be viable. With these additions, the system could then be used to assist in the control of alien invasive plants, providing a useful tool to assist with both its monitoring and management and hopefully to assist in its effective control.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. 2016. *TensorFlow: Large-scale machine learning on heterogeneous systems.* Software available from tensorflow.org.

Abd-Elrahman, A., Pearlstine, L., & Percival, F. 2005. Development of pattern recognition algorithm for automatic bird detection from unmanned aerial vehicle imagery. *Surveying and Land Information Science*, **65**(1), 37.

Abouzahir, S., Sadik, M., & Sabir, E. 2017. IoT- empowered smart agriculture: a real-time light-weight embedded segmentation system. *Pages 319–332 of:* Sabir, E., García Armada, A., Ghogho, M., & Debbah, M. (eds), *International Symposium on Ubiquitous Networking.* Springer.

Affouard, A., Goëau, H., Bonnet, P., Lombardo, J.-C., & Joly, A. 2017 (April). Pl@ntNet app in the era of deep learning. *Pages 1–6 of: 5th International Conference on Learning Representations.*

Ajayi, O., Salubi, A., Angbas, A., & Odigure, M. 2017. Generation of accurate digital elevation models from UAV acquired low percentage overlapping images. *International Journal of Remote Sensing*, **38**(8-10), 3113–3134.

Alexe, B., Deselaers, T., & Ferrari, V. 2012. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**(11), 2189–2202.

Alvarez-Taboada, F., Paredes, C., & Julián-Pelaz, J. 2017. Mapping of the invasive species

Hakea sericea using Unmanned Aerial Vehicle (UAV) and WorldView-2 imagery and an object-oriented approach. *Remote Sensing*, **9**(9).

Anar, A. C., Bostanci, E., & Guzel, M. S. 2018. Live target detection with deep learning neural network and unmanned aerial vehicle on Android mobile device. *arXiv preprint arXiv:1803.07015*.

Andelson, E., Anderson, C., Bergen, J., Burt, P., & Ogden, J. 1984. Pyramid methods in image processing. *RCA Engineer*, **29**(6), 33–41.

Baron, J., Hill, D., & Elmiligi, H. 2018. Combining image processing and machine learning to identify invasive plants in high-resolution images. *International Journal of Remote Sensing*, **39**(15–16), 5099–5118.

Bischke, B., Helber, P., Borth, D., & Dengel, A. 2018. Segmentation of imbalanced classes in satellite imagery using adaptive uncertainty weighted class loss. *IEEE International Geoscience and Remote Sensing Symposium 2018*, 6191–6194.

Blaschke, T. 2010. Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, **65**(1), 2–16.

Bouchareb, Y. 2018. *Machine learning applied to aerial images for vegetation detection.* Poster presented at: The Deep Learning Indaba 2018, Stellenbosch, SA.

Bradley, B. 2014. Remote detection of invasive plants: a review of spectral, textural and phenological approaches. *Biological Invasions*, **16**(7), 1411–1425.

Buch, A., & Dixon, A. B. 2009. South Africa's Working for Water programme: searching for win–win outcomes for people and the environment. *Sustainable Development*, **17**(3), 129–141.

Burdziakowski, P. 2017. Evaluation of Open Drone Map toolkit for geodetic grade aerial drone mapping–case study. *Pages 101–110 of: 17th International Multidisciplinary Scientific GeoConference*, vol. 17.

Calleja, F., Ondiviela, B., Galván, C., Recio, M., & Juanes, J. 2019. Mapping estuarine vegetation using satellite imagery: The case of the invasive species *Baccharis halimifolia* at a Natura 2000 site. *Continental Shelf Research*, **174**, 35–47.

Camargo, N. J. 2004. *A combined statistical-soft computing approach for classification and mapping weed species in minimum-tillage systems.* Ph.D. thesis, The University of Nebraska - Lincoln.

Carreira, J., & Sminchisescu, C. 2010. Constrained parametric min-cuts for automatic object segmentation. *Pages 3241–3248 of: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE.

Carrio, A., Sampedro, C., Rodriguez-Ramos, A., & Campoy, P. 2017. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, **2017**.

Chabot, D., & Bird, D. 2015. Wildlife research and management methods in the 21st century: Where do unmanned aircraft fit in? *Journal of Unmanned Vehicle Systems*, **3**(4), 137–155.

Chabot, D., & Bird, D. M. 2012. Evaluation of an off-the-shelf unmanned aircraft system for surveying flocks of geese. *Waterbirds*, **35**(1), 170–175.

Chabot, D., Dillon, C., Shemrock, A., Weissflog, N., & Sager, E. 2018. An object-based image analysis workflow for monitoring shallow-water aquatic vegetation in multispectral drone imagery. *ISPRS International Journal of Geo-Information*, **7**(8), 294.

Chao, H., Cao, Y., & Chen, Y. 2010. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, **8**(1), 36–44.

Cheng, M., Zhang, Z., Lin, W., & Torr, P. 2014. BING: Binarized normed gradients for objectness estimation at 300fps. *Pages 3286–3293 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Christiansen, P., Steen, K., Jørgensen, R., & Karstoft, H. 2014. Automated detection and recognition of wildlife using thermal cameras. *Sensors*, **14**(8), 13778–13793.

Clouard, R., Renouf, A., & Revenu, M. 2010. An ontology-based model for representing image processing objectives. *International Journal of Pattern Recognition and Artificial Intelligence*, **24**(08), 1181–1208.

Colomina, I., & Molina, P. 2014. Unmanned aerial systems for photogrammetry and remote sensing: a review. *ISPRS Journal of Photogrammetry and Remote Sensing*, **92**, 79–97.

Cruzan, M., Weinstein, B., Grasty, M., Kohrn, B., Hendrickson, E., Arredondo, T., & Thompson, P. 2016. Small unmanned aerial vehicles (micro-UAVs, drones) in plant ecology. *Applications in Plant Sciences*, **4**(9).

de Sá, N. C., Castro, P., Carvalho, S., Marchante, E., López-Núñez, F. A., & Marchante,

H. 2018. Mapping the flowering of an invasive plant using unmanned aerial vehicles: is there potential for biocontrol monitoring? *Frontiers in Plant Science*, **9**, 293.

Department of Water Affairs. 1999. *Working for Water.* `http://www.dwaf.gov.za/wfw/default.aspx`. Accessed on: 14/10/2017.

DJI. 2017. *Phantom 3 Professional.* `http://www.dji.com/phantom-3-pro`. Accessed on: 08/11/2017.

DJI. 2019. *DJI GO.* `https://www.dji.com/goapp`. Accessed on: 12/3/2019.

DJI Support. 2017 (October). *DJI GO 4 Manual: The Pilot's Handbook.* `https://store.dji.com/guides/dji-go-4-manual/8/`.

Downey, P. O., & Richardson, D. M. 2016. Alien plant invasions and native plant extinctions: a six-threshold framework. *AoB Plants*, **8**.

Dvorák, P., Müllerová, J., Bartalos, T., & Bruna, J. 2015. Unmanned aerial vehicles for alien plant species detection and monitoring. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, **40**(1), 83–90.

Endres, I., & Hoiem, D. 2010. Category independent object proposals. *Pages 575–588 of: European Conference on Computer Vision.* Springer.

Felzenszwalb, P., Girshick, R., & McAllester, D. 2010a. Cascade object detection with deformable part models. *Pages 2241–2248 of: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE.

Felzenszwalb, P., Girshick, R., McAllester, D., & Ramanan, D. 2010b. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(9), 1627–1645.

Floreano, D., & Wood, R. 2015. Science, technology and the future of small autonomous drones. *Nature*, **521**(7553), 460–466.

Franco, C., Guada, C., Rodríguez, J. T., Nielsen, J., Rasmussen, J., Gómez, D., & Montero, J. 2018. Automatic detection of thistle-weeds in cereal crops from aerial RGB images. *Pages 441–452 of: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems.* Springer.

Ghazbi, S. N., Aghli, Y., Alimohammadi, M., & Akbari, A. A. 2016. Quadrotors unmanned aerial vehicles: a review. *International Journal on Smart Sensing and Intelligent Systems*, **9**, 309–333.

Gillespie, A. R., Kahle, A. B., & Walker, R. E. 1987. Color enhancement of highly correlated images. II. Channel ratio and "chromaticity" transformation techniques. *Remote Sensing of Environment*, **22**(3), 343–365.

Girshick, R. 2015. Fast R-CNN. *Pages 1440–1448 of: Proceedings of the IEEE International Conference on Computer Vision*. IEEE.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Pages 580–587 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Gitelson, A. A., Kaufman, Y. J., Stark, R., & Rundquist, D. 2002. Novel algorithms for remote estimation of vegetation fraction. *Remote Sensing of Environment*, **80**(1), 76–87.

Goëau, H., Bonnet, P., Joly, A., Bakić, V., Barbe, J., Yahiaoui, I., Selmi, S., Carré, J., Barthélémy, D., Boujemaa, N., *et al.* 2013. Pl@ntnet mobile app. *Pages 423–424 of: Proceedings of the 21st ACM International Conference on Multimedia*. ACM.

Göktogan, A., Sukkarieh, S., Bryson, M., Randle, J., Lupton, T., & Hung, C. 2010. A rotary-wing unmanned air vehicle for aquatic weed surveillance and management. *Journal of Intelligent and Robotic Systems*, **57**(1), 467–484.

Grenzdörffer, G. 2013. UAS-based automatic bird count of a common gull colony. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **XL-1/W2**, 169–174.

Grm, K., Štruc, V., Artiges, A., Caron, M., & Ekenel, H. K. 2018. Strengths and weaknesses of deep learning models for face recognition against image degradations. *IET Biometrics*, **7**(1), 81–89.

Guerrero, J. M., Pajares, G., Montalvo, M., Romeo, J., & Guijarro, M. 2012. Support vector machines for crop/weeds identification in maize fields. *Expert Systems with Applications*, **39**(12), 11149–11155.

Guijarro, M., Pajares, G., Riomoros, I., Herrera, P., Burgos-Artizzu, X., & Ribeiro, A. 2011. Automatic segmentation of relevant textures in agricultural images. *Computers and Electronics in Agriculture*, **75**(1), 75–83.

Guirado, E., Tabik, S., Alcaraz-Segura, D., Cabello, J., & Herrera, F. 2017. Deep-learning versus OBIA for scattered shrub detection with Google Earth imagery: *Ziziphus Lotus* as case study. *Remote Sensing*, **9**(12).

Gupta, S. G., Ghonge, M. M., & Jawandhiya, P. 2013. Review of unmanned aircraft system (UAS). *International Journal of Advanced Research in Computer Engineering &Technology*, **2**(4), 1646–1658.

Habel, J., Teucher, M., Ulrich, W., & Schmitt, T. 2018. Documenting the chronology of ecosystem health erosion along East African rivers. *Remote Sensing in Ecology and Conservation*, **4**(1), 34–43.

Hague, T., Tillett, N., & Wheeler, H. 2006. Automated crop and weed monitoring in widely spaced cereals. *Precision Agriculture*, **7**(1), 21–32.

Hamuda, E., Glavin, M., & Jones, E. 2016. A survey of image processing techniques for plant extraction and segmentation in the field. *Computers and Electronics in Agriculture*, **125**, 184–199.

Harris Geospatial Solutions. 2017. *Vegetation indices background.* `http://www.harrisgeospatial.com/docs/backgroundvegetationindices.html`. Accessed on: 5/10/2017.

He, K., Zhang, X., Ren, S., & Sun, J. 2016. Deep residual learning for image recognition. *Pages 770–778 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Hill, D., Tarasoff, C., Whitworth, G., Baron, J., Bradshaw, J., & Church, J. 2017. Utility of unmanned aerial vehicles for mapping invasive plant species: a case study on yellow flag iris (*Iris pseudacorus* L.). *International Journal of Remote Sensing*, **38**(8-10), 2083–2105.

Hodgeson, H. 2013. Precision herbicide drones launch strikes on weeds. *Newscientist*, 19.

Hodgson, J. C., Mott, R., Baylis, S. M., Pham, T. T., Wotherspoon, S., Kilpatrick, A. D., Raja Segaran, R., Reid, I., Terauds, A., & Koh, L. P. 2018. Drones count wildlife more accurately and precisely than humans. *Methods in Ecology and Evolution*, **9**(5), 1160–1167.

Hong, S.-J., Han, Y., Kim, S.-Y., Lee, A.-Y., & Kim, G. 2019. Application of deep-learning methods to bird detection using unmanned aerial vehicle imagery. *Sensors*, **19**(7), 1651.

Hosang, J., Benenson, R., & Schiele, B. 2014. How good are detection proposals, really? *In: Proceedings of the British Machine Vision Conference*. BMVA Press.

Hough, P. V. 1962 (Dec. 18). *Method and means for recognizing complex patterns.* US Patent 3 069 654.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861.*

Huang, C., & Asner, G. 2009. Applications of remote sensing to alien invasive plant studies. *Sensors*, **9**(6), 4869–4889.

Huete, A., Didan, K., Miura, T., Rodriguez, E. P., Gao, X., & Ferreira, L. G. 2002. Overview of the radiometric and biophysical performance of the MODIS vegetation indices. *Remote Sensing of Environment*, **83**(1-2), 195–213.

Hung, C., Xu, Z., & Sukkarieh, S. 2014. Feature learning based approach for weed classification using high resolution aerial images from a digital camera mounted on a UAV. *Remote Sensing*, **6**(12), 12037–12054.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5 MB model size. *arXiv preprint arXiv:1602.07360.*

James, K., & Bradshaw, K. 2019. Segmenting objects with indistinct edges, with application to aerial imagery of vegetation. *In: Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists.* ACM.

Jiménez López, J., & Mulero-Pázmány, M. 2019. Drones for conservation in protected areas: present and future. *Drones*, **3**(1).

Kamilaris, A., & Prenafeta-Boldu, F. X. 2018. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, **147**, 70–90.

Kataoka, T., Kaneko, T., Okamoto, H., & Hata, S. 2003. Crop growth estimation system using machine vision. *Pages b1079–b1083 of: Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol. 2. IEEE.

Khan, M. 2014. A survey: image segmentation techniques. *International Journal of Future Computer and Communication*, **3**(2), 89–93.

Khan, S., & Madden, M. 2009. A survey of recent trends in one class classification. *Pages 188–197 of: Irish Conference on Artificial Intelligence and Cognitive Science.* Springer.

Kluge, R., & De Beer, H. 1984. *Silky Hakea.* Technical report. Weeds and Pesticides Subdivision Plant Protection Research Institute.

Koh, L., & Wich, S. 2012. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. *Tropical Conservation Science*, **5**(2), 121–132.

Kohavi, R. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Pages 1137–1145 of: The International Joint Conference on Artificial Intelligence*, vol. 14.

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. 2007. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, **160**, 3–24.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. *Pages 1097–1105 of:* Pereira, F., Burges, C. J. C., Bottou, L., & Weinberger, K. Q. (eds), *Advances in Neural Information Processing Systems 25.* Curran Associates, Inc.

Kumar, K., & Jaspreet, S. 2017. Analysis of Image Enhancement Algorithms. *I-manager's Journal on Information Technology*, **6**(1), 25–35.

Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., & Soares, J. V. 2012. Leafsnap: A computer vision system for automatic plant species identification. *Pages 502–516 of: European Conference on Computer Vision 2012.* Springer.

Kung, S. 2014. *Kernel methods and machine learning.* Cambridge University Press.

Le Maitre, D. C., Forsyth, G. G., Dzikiti, S., & Gush, M. B. 2016. Estimates of the impacts of invasive alien plants on water flows in South Africa. *Water SA*, **42**(4), 659–672.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. 1990. Handwritten digit recognition with a back-propagation network. *Pages 396–404 of: Advances in Neural Information Processing Systems.*

Lehmann, J., Prinz, T., Ziller, S., Thiele, J., Heringer, G., Meira-Neto, J., & Buttschardt, T. 2017. Open-source processing and analysis of aerial imagery acquired with a low-cost unmanned aerial system to support invasive plant management. *Frontiers in Environmental Science*, **5**(44).

Li, R., Liu, W., Yang, L., Sun, S., Hu, W., Zhang, F., & Li, W. 2018. DeepUNet: A deep fully convolutional network for pixel-level sea-land segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, **11**(11), 3954–3962.

Lin, T., Goyal, P., Girshick, R., He, K., & Dollar, P. 2017. Focal loss for dense object detection. *Pages 2999–3007 of: 2017 IEEE International Conference on Computer Vision*. IEEE.

Liu, T., Abd-Elrahman, A., Jon, M., & Wilhelm, V. 2018. Comparing fully convolutional networks, random forest, support vector machine, and patch-based deep convolutional neural networks for object-based wetland mapping using images from small unmanned aircraft system. *GIScience & Remote Sensing*, **55**(2), 243–264.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. 2016. SSD: single shot multibox detector. *Pages 21–37 of: European Conference on Computer Vision*. Springer.

Long, J., Shelhamer, E., & Darrell, T. 2015. Fully convolutional networks for semantic segmentation. *Pages 3431–3440 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Lopez-Granados, F. 2011. Weed detection for site-specific weed management: mapping and real-time approaches. *Weed Research*, **51**(1), 1–11.

Louhaichi, M., Borman, M. M., & Johnson, D. E. 2001. Spatially located platform and aerial photography for documentation of grazing impacts on wheat. *Geocarto International*, **16**(1), 65–70.

Mafanya, M., Tsele, P., Botai, J., Manyama, P., Swart, B., & Monate, T. 2017. Evaluating pixel and object based image classification techniques for mapping plant invasions from UAV derived aerial imagery: *Harrisia pomanensis* as a case study. *ISPRS Journal of Photogrammetry and Remote Sensing*, **129**, 1–11.

Maire, F., Mejias, L., Hodgson, A., & Duclos, G. 2013 (11). Detection of dugongs from unmanned aerial vehicles. *Pages 2750–2756 of: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.

Maire, F., Alvarez, L. M., & Hodgson, A. 2015. Automating marine mammal detection in aerial images captured during wildlife surveys: a deep learning approach. *Pages 379–385 of: Australasian Joint Conference on Artificial Intelligence*. Springer.

Malisiewicz, T., Shrivastava, A., Gupta, A., & Efros, A. 2012. Exemplar-SVMs for visual

object detection, label transfer and image retrieval. *In: International Conference on Machine Learning.*

Manen, S., Guillaumin, M., & Van Gool, L. 2013. Prime object proposals with randomized Prim's algorithm. *Pages 2536–2543 of: Proceedings of the IEEE International Conference on Computer Vision.* IEEE.

Marris, E. 2013. Drones in science: fly, and bring me data. *Nature*, **498**(7453), 156–158.

Martin, F.-M., Müllerová, J., Borgniet, L., Dommanget, F., Breton, V., & Evette, A. 2018. Using single- and multi-date UAV and satellite imagery to accurately monitor invasive knotweed species. *Remote Sensing*, **10**(10).

Mboga, N., Georganos, S., Grippa, T., Lennert, M., Vanhuysse, S., & Wolff, E. 2018 (June). Fully convolutional networks for the classification of aerial VHR imagery. *Pages 1–12 of: Proceedings of the 7th Geographic Object-Based Image Analysis (GEOBIA) Conference.*

Milioto, A., Lottes, P., & Stachniss, C. 2017. Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **IV-2/W3**, 41–48.

Milletari, F., Navab, N., & Ahmadi, S.-A. 2016 (10). V-Net: fully convolutional neural networks for volumetric medical image segmentation. *Pages 565–571 of: 2016 Fourth International Conference on 3D Vision.* IEEE.

Monteiro, A., Von Wangenheim, A., & Júnior, P. 2019 (05). *Weed Mapping on Aerial Images: A Systematic Literature Review.* Technical report. INCoD/LAPIX.01.2019.E. Brazilian Institute for Digital Convergence.

Müllerová, J., Bartaloš, T., Bråna, J., Dvořák, P., & Vítková, M. 2017. Unmanned aircraft in nature conservation: an example from plant invasions. *International Journal of Remote Sensing*, **38**(8-10), 2177–2198.

Nahar, P., Wu, K.-h., Mei, S., Ghoghari, H., Srinivasan, P., Lee, Y.-l., Gao, J., & Guan, X. 2017. Autonomous UAV forced graffiti detection and removal system based on machine learning. *Pages 1–8 of: 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation.* IEEE.

Nex, F., & Remondino, F. 2014. UAV for 3D mapping applications: a review. *Applied Geomatics*, **6**(1), 1–15.

Nogueira, K., Penatti, O. A., & dos Santos, J. A. 2016. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, **61**, 539–556.

Nowak, M., Bogawski, P., & DziÃ³b, K. 2019. Unmanned aerial vehicles (UAVs) in environmental biology: a review. *European Journal of Ecology*, **4**(03), 56–74.

Olivares-Mendez, M., Fu, C., Ludivig, P., Bissyandé, T., Kannan, S., Zurad, M., Annaiyan, A., Voos, H., & Campoy, P. 2015. Towards an autonomous vision-based unmanned aerial system against wildlife poachers. *Sensors*, **15**(12), 31362–31391.

Onishi, M., & Ise, T. 2018. Automatic classification of trees using a UAV onboard camera and deep learning. *arXiv preprint arXiv:1804.10390*.

OpenCV. 2015. *Discriminatively trained part based models for object detection*. `https://docs.opencv.org/3.1.0/d9/d12/group__dpm.html`. Accessed on: 31/10/2017.

OpenCV. 2017. *Cascade classification: Haar feature-based cascade classifier for object detection*. `https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html`. Accessed on: 30/10/2017.

OpenCV team. 2019. *OpenCV about*. `https://opencv.org/about.html`. Accessed on: 12/3/2019.

Osman, H. 2010. Random forest-LNS architecture and vision. *Chap. 10 of:* Zhang, Y. (ed), *New Advances in Machine Learning*. InTechOpen.

Pajares, G. 2015. Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs). *Photogrammetric Engineering & Remote Sensing*, **81**(4), 281–330.

Palmer, T. 2004. *Vegetation of Makana*. Technical report. ARC - Range and Forage Institute, Grahamstown.

Patton, F. 2013. Are drones the answer to the conservationist's prayers? *SWARA*, **July–September**, 52–55.

Peña, J., Torres-Sánchez, J., de Castro, A., Kelly, M., & López-Granados, F. 2013. Weed mapping in early-season maize fields using object-based analysis of unmanned aerial vehicle (UAV) images. *PLOS One*, **8**(10).

Pérez-Ortiz, M., Pena, J., Gutiérrez, P., Torres-Sánchez, J., Hervás-Martínez, C., & López-Granados, F. 2015. A semi-supervised system for weed mapping in sunflower crops using unmanned aerial vehicles and a crop row detection method. *Applied Soft Computing*, **37**, 533–544.

Pimentel, D., Zuniga, R., & Morrison, D. 2005. Update on the environmental and economic costs associated with alien-invasive species in the United States. *Ecological Economics*, **52**(3), 273–288.

Prasvita, D. S., & Herdiyeni, Y. 2013. MedLeaf: mobile application for medicinal plant identification based on leaf image. *International Journal on Advanced Science, Engineering and Information Technology*, **3**(2), 103–106.

Rahman, M. A., & Wang, Y. 2016. Optimizing intersection-over-union in deep neural networks for image segmentation. *Pages 234–244 of: International Symposium on Visual Computing*. Springer.

Rahtu, E., Kannala, J., & Blaschko, M. 2011. Learning a category independent object detection cascade. *Pages 1052–1059 of: Proceedings of the IEEE International Conference on Computer Vision*. IEEE.

Rango, A., Laliberte, A., Herrick, J., Winters, C., Havstad, K., Steele, C., & Browning, D. 2009. Unmanned aerial vehicle-based remote sensing for rangeland assessment, monitoring, and management. *Journal of Applied Remote Sensing*, **3**(1).

Rantalankila, P., Kannala, J., & Rahtu, E. 2014. Generating object segmentation proposals using global and local search. *Pages 2417–2424 of: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Raschka, S., & Mirjalili, V. 2017. *Python machine learning: machine learning and deep learning with Python, Scikit-Learn, and Tensorflow*. Packt.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016. You Only Look Once: unified, real-time object detection. *Pages 779–788 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Remondino, F., Barazzetti, L., Nex, F., Scaioni, M., & Sarazzi, D. 2011. UAV photogrammetry for mapping and 3d modeling–current status and future perspectives. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **XXXVIII-1/C22**, 25–31.

Ren, S., He, K., Girshick, R., & Sun, J. 2015. Faster R-CNN: Towards real-time ob-

ject detection with region proposal networks. *Pages 91–99 of:* Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., & Garnett, R. (eds), *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Rey, N., Volpi, M., Joost, S., & Tuia, D. 2017. Detecting animals in African Savanna with UAVs and the crowds. *Remote Sensing of Environment*, **200**, 341–351.

Richardson, D. M., & Van Wilgen, B. W. 2004. Invasive alien plants in South Africa: how well do we understand the ecological impacts? *South African Journal of Science*, **100**(1-2), 45–52.

Richardson, D., Van Wilgen, B., & Mitchell, D. 1987. Aspects of the reproductive ecology of four Australian Hakea species (Proteaceae) in South Africa. *Oecologia*, **71**(3), 345–354.

Rominger, K., & Meyer, S. E. 2019. Application of UAV-based methodology for census of an endangered plant species in a fragile habitat. *Remote Sensing*, **11**(6).

Ronneberger, O., Fischer, P., & Brox, T. 2015. U-Net: Convolutional networks for biomedical image segmentation. *Pages 234–241 of: International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer.

Rouse Jr, J., Haas, R., Schell, J., & Deering, D. 1974. Monitoring vegetation systems in the Great Plains with ERTS. *Pages 309–317 of: Proceedings of the Third Earth Resources Technology Satellite - 1 Symposium*, vol. 1. NASA.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, **115**(3), 211–252.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks*, **61**, 85–117.

Schowengerdt, R. 2007. *Remore sensing: models and methods for image processing*. 3 edn. Elsevier.

Selby, W., Corke, P., & Rus, D. 2011. Autonomous aerial navigation and tracking of marine animals. *Pages 1–7 of: Proceedings of the Australian Conference on Robotics and Automation*. Australian Robotics & Automation Association.

Seymour, A., Dale, J., Hammill, M., Halpin, P., & Johnston, D. 2017. Automated detec-

tion and enumeration of marine wildlife using unmanned aircraft systems (UAS) and thermal imagery. *Scientific Reports*, **7**, 45127.

Shiferaw, H., Bewket, W., & Eckert, S. 2019. Performances of machine learning algorithms for mapping fractional cover of an invasive plant species in a dryland ecosystem. *Ecology and Evolution*, **9**(5), 2562–2574.

Simonyan, K., & Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*

Sonka, M., Hlavac, V., & Boyle, R. 2014. *Image processing, analysis, and machine vision.* second edn. Cengage Learning.

Stumph, B., Virto, M. H., Medeiros, H., Tabb, A., Wolford, S., Rice, K., & Leskey, T. 2019. Detecting invasive insects with unmanned aerial vehicles. *arXiv preprint arXiv:1903.00815.*

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Cardoso, M. J. 2017. Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. *Pages 240–248 of: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support.* Springer.

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

Torres-Sánchez, J., López-Granados, F., De Castro, A., & Peña-Barragán, J. 2013. Configuration and specifications of an unmanned aerial vehicle (UAV) for early site specific weed management. *PLOS ONE*, **8**(3), 1–15.

Torres-Sánchez, J., Peña, J., De Castro, A., & López-Granados, F. 2014. Multi-temporal mapping of the vegetation fraction in early-season wheat fields using images from UAV. *Computers and Electronics in Agriculture*, **103**, 104–113.

Tucker, C. J. 1979. Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, **8**(2), 127–150.

Uijlings, J., Van De Sande, K., Gevers, T., & Smeulders, A. 2013. Selective search for object recognition. *International Journal of Computer Vision*, **104**(2), 154–171.

Vale, G. 2015. *Raspberry Drone: Unmanned Aerial Vehicle.* Masters thesis, Instituto Superior Técnico, Lisboa, Portugal.

Van Andel, A., Wich, S., Boesch, C., Koh, L., Robbins, M., Kelly, J., & Kuehl, H. 2015. Locating chimpanzee nests and identifying fruiting trees with an unmanned aerial vehicle. *American Journal of Primatology*, **77**(10), 1122–1134.

Van Gemert, J., Verschoor, C., Mettes, P., Epema, K., Koh, L., & Wich, S. 2015. Nature conservation drones for automatic localization and counting of animals. *Pages 255–270 of: European Conference on Computer Vision 2014 Workshops*. Cham: Springer International Publishing.

Vedaldi, A., Gulshan, V., Varma, M., & Zisserman, A. 2009. Multiple kernels for object detection. *Pages 606–613 of: 2009 IEEE 12th International Conference on Computer Vision*. IEEE.

Ventura, D., Bonifazi, A., Gravina, M., Belluscio, A., & Ardizzone, G. 2018. Mapping and classification of ecologically sensitive marine habitats using unmanned aerial vehicle (UAV) imagery and Object-Based Image Analysis (OBIA). *Remote Sensing*, **10**(9).

Vondrick, C., Patterson, D., & Ramanan, D. 2013. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, **101**(1), 184–204.

Wäldchen, J., Rzanny, M., Seeland, M., & Mäder, P. 2018. Automated plant species identification - Trends and future directions. *PLOS Computational Biology*, **14**(4), 1–19.

Whitehead, K., & Hugenholtz, C. 2014. Remote sensing of the environment with small unmanned aircraft systems (UASs), part 1: a review of progress and challenges. *Journal of Unmanned Vehicle Systems*, **2**(3), 69–85.

Winberg, S., Ramone, M., & Naidoo, K. 2017. Accelerating computer-based recognition of fynbos leaves using a graphics processing unit. *South African Computer Journal*, **29**(3), 238–262.

Woebbecke, D. M., Meyer, G. E., Von Bargen, K., & Mortensen, D. 1995. Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the American Society of Agricultural Engineers*, **38**(1), 259–269.

Xiao, X.-Y., Hu, R., Zhang, S.-W., & Wang, X.-F. 2010. HOG-based approach for leaf classification. *Pages 149–155 of: International Conference on Intelligent Computing*. Springer.

Yao, W., Poleswki, P., & Krzystek, P. 2016. Classification of urban aerial data based on pixel labelling with deep convolutional neural networks and logistic regression. *IS-*

PRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, **XLI-B7**, 405–410.

Yi, S. 2016. FragMAP: a tool for long-term and cooperative monitoring and analysis of small-scale habitat fragmentation using an unmanned aerial vehicle. *International Journal of Remote Sensing*, **38**(8-10), 2686–2697.

Yu, H., & Kim, S. 2012. SVM tutorial – classification, regression and ranking. *Pages 479–506 of: Handbook of Natural Computing*. Springer.

Zeiler, M. D., & Fergus, R. 2014. Visualizing and understanding convolutional networks. *Pages 818–833 of: European Conference on Computer Vision*. Cham: Springer International Publishing.

Zhang, L., Zhang, L., & Du, B. 2016. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, **4**(2), 22–40.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. 2016. Learning deep features for discriminative localization. *Pages 2921–2929 of: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Zhou, Z. 2012. *Ensemble methods: foundations and algorithms*. Machine Learning and Pattern Recognition. Chapman & Hall/CRC.

Zitnick, C., & Dollár, P. 2014. Edge boxes: Locating object proposals from edges. *Pages 391–405 of: European Conference on Computer Vision*. Springer.

Appendices

Appendix A

Landowner permission forms

27 January 2018
Mr Rodney Tyson
Springfield farm
Grahamstown

Dear Mr Rodney Tyson,

I am writing to request permission to conduct a research study on your land. I am currently enrolled in the computer science master's program at Rhodes University and am in the process of writing my dissertation, titled "Technology in conservation: a framework for drone control of invasive vegetation."

In this research I will be particularly focusing on the *Hakea* species, an invasive shrub from Australia. To conduct this research it is necessary that I fly a drone, in particular a DJI Mavic Pro quadcopter, over an area of land with the appropriate vegetation type. While flying, the drone will use its onboard camera to take photographs and videos of the vegetation below and record the GPS locations of the plants. This data will be used to develop an artificial intelligence system capable of automatically detecting *Hakea* plants. Only photographs of vegetation will appear in my dissertation and no personal objects that may be in breach of your privacy will be photographed.

I hope to have completed my research by December 2018. Your approval to conduct this research study on your land would be greatly appreciated. I would be happy to answer any questions or concerns that you may have. You may contact me on my email address g13j0359@campus.ru.ac.za. My supervisor, Karen Bradshaw, may be contacted on her email address k.bradshaw@ru.ac.za.

If you give your consent, kindly sign below and I will collect this form from you when convenient.

Sincerely,

Katherine M James

Katherine James (Rhodes University)

Consent given by:

RODNEY TYSON	R Tyson	30/1/2018.
Print your name here	Signature	Date

08/02/2019

Springfield Farm

Dear Mr Rodney Tyson,

Research update and request for continuation of permission to fly drone

I am writing to request your continued permission to conduct a research study on your land in 2019. I am currently enrolled in my final year of the computer science master's program at Rhodes University and am in the process of writing my dissertation, titled "Technology in conservation: a framework for drone control of invasive vegetation."

In this research I am particularly focusing on the *Hakea* species, an invasive shrub from Australia. To conduct this research it is necessary that I fly a drone, in particular a DJI Mavic Pro quadcopter, over an area of land with the appropriate vegetation type. While flying, the drone will use its onboard camera to take photographs and videos of the vegetation.

I have recently completed the section of my thesis in which I have developed an artificial intelligence system capable of automatically detecting *Hakea* shrubs. The next stage of my thesis involves testing what accuracy is obtainable when testing this system in the field.

Your approval to conduct this research study on your land would be greatly appreciated. I would be happy to answer any questions or concerns that you may have. You may contact me on my email address g13j0359@campus.ru.ac.za. My supervisor, Professor Karen Bradshaw, is contactable via the address k.bradshaw@ru.ac.za.

If you give your consent to the continuation of my study on your land, kindly sign below and I will collect this form from you when convenient.

Sincerely,

Katherine James

MSc candidate at Rhodes University

Consent given by:

Contact number/email:

27 January 2018
Dr Stuart Dwyer
Mountain View Farm
Grahamstown

Dear Dr Stuart Dwyer,

I am writing to request permission to conduct a research study on your land. I am currently enrolled in the computer science master's program at Rhodes University and am in the process of writing my dissertation, titled "Technology in conservation: a framework for drone control of invasive vegetation."

In this research I will be particularly focusing on the *Hakea* species, an invasive shrub from Australia. To conduct this research it is necessary that I fly a drone, in particular a DJI Mavic Pro quadcopter, over an area of land with the appropriate vegetation type. While flying, the drone will use its onboard camera to take photographs and videos of the vegetation below and record the GPS locations of the plants. This data will be used to develop an artificial intelligence system capable of automatically detecting *Hakea* plants. Only photographs of vegetation will appear in my dissertation and no personal objects that may be in breach of your privacy will be photographed.

I hope to have completed my research by December 2018. Your approval to conduct this research study on your land would be greatly appreciated. I would be happy to answer any questions or concerns that you may have. You may contact me on my email address g13j0359@campus.ru.ac.za. My supervisor, Karen Bradshaw, may be contacted on her email address k.bradshaw@ru.ac.za.

If you give your consent, kindly sign below and I will collect this form from you when convenient.

Sincerely,

Katherine M James

Katherine James (Rhodes University)

Consent given by:

DWyer

Anne Dwyer		30/01/2018
Print your name here	Signature	Date

2018/1/30