University of Cape Town

Faculty of Science

Department of Statistical Sciences

Masters Half Dissertation

Detection and Isolation of Prey Capture Events in Animal-Borne Images

Author:
Temweka S. Chirwa

Supervisor:
A/Prof Ian Durbach
Co-Supervisor:
Dr Emmanuel Dufourq

Centre for Statistics in Ecology, the Environment, and Conservation

March 2021

Declaration of Authorship

I, Temweka Chirwa, declare that this thesis titled, "Detection and Isolation of Prey Capture Events in Animal-Borne Images" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Name: Temweka Chirwa

Signature: Signed by candidate

Date: 16/10/2021

Abstract

Understanding the foraging habits and prey availability for a species is crucial. Prey availability is crucial to a species' survival and sustainability of the food pyramid. Identifying the type of prey consumed also allows ecologists to determine the energy received, while the duration and extent of foraging bouts provide information about the energy expended.

With recent advancements in technology, data collection has become more accessible, and animal-borne video cameras are an increasingly popular mechanism for collecting information about foraging and other behaviour. Video recorders collect large volumes of data but create a bottleneck as data processing is still predominantly done manually. This process is time-consuming and costly, even with the assistance of crowdsourced tasks.

Advancements in deep learning, and its applications to computer vision, provide opportunities to apply these tools to ecological problems, such as the processing of data from animal-borne video recorders. Speeding up the annotation process allows more time to be spent focused on the ecological research questions.

This dissertation aims to develop detection and isolation models that will assist in the processing of visual data, namely images from animal-borne videos. The first model used for detection will perform an image classification determining whether prey is present or not. Images found to have prey present will then be presented to the second model for isolation that identifies exactly where within the image the prey is and labels the type of prey. The models were trained on video data of little penguins (*Eudyptula minor*), whose main prey in this investigation are small fish, predominantly anchovies, and jellyfish.

The image classification model based on the ResNet architecture achieved 85% accuracy with precision and recall values of 0.85 and 0.85 respectively on its test set. The object detection model based on the You Only Look Once (YOLO) framework achieved a mean average precision of 60% on its test set. However, the models did not perform well enough on unseen full length videos to be used without human supervision or to serve as alternatives to manual labelling. Rather, the models can be used to guide researchers to areas that may contain prey events.

Acknowledgements

I would like to thank my supervisors, Associate Professor Ian Durbach, Dr Emmanuel Dufourq and Associate Professor Francesca Little, for their kindness, patience, expertise and general commitment throughout this process. Thank you.

I would like to acknowledge Dr John Arnould and Dr Grace Sutton for availing their dataset to be used in this dissertation.

I would also like to thank my family and friends for their endless love, support and encouragement during this period. I would not have gotten here without you all. Above all else, I would like to thank the Almighty Father, through whom all things are possible.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Research Objectives . . . 2
- 1.3 Little Penguins . . . 3
- 1.4 Outline of the Dissertation . . . 4

2 Basics of Neural Networks 5
- 2.1 Artificial Intelligence and Machine Learning . . . 5
- 2.2 Feed Forward Neural Networks . . . 6
- 2.3 Activation Functions . . . 8
- 2.4 Loss Functions . . . 9
  - 2.4.1 Cross Entropy Loss Function . . . 10
- 2.5 Optimization . . . 10
  - 2.5.1 Backpropagation . . . 10
  - 2.5.2 Optimization Functions . . . 11
- 2.6 Generalization . . . 12
  - 2.6.1 Weight Regularization . . . 12
  - 2.6.2 Early Stopping . . . 12
  - 2.6.3 Dropout . . . 13
- 2.7 Hyperparameter Optimization . . . 13
- 2.8 Deep Learning and Transfer Learning . . . 13

3 Computer Vision 16
- 3.1 Convolutional Neural Networks (CNNs) . . . 16
- 3.2 Image Classification . . . 18
  - 3.2.1 Model Evaluation . . . 19
- 3.3 Object Detection . . . 20
  - 3.3.1 Model Evaluation . . . 23
- 3.4 Deep Learning in Marine or Ecological projects . . . 25

4 Methodology **26**
4.1 Data Description 26
4.2 Data Extraction 26
4.3 Image Classification 27
4.3.1 Pre-processing Methods 27
4.3.2 Neural Network Methods 28
4.3.3 Hyperparameters 29
4.3.4 Software Packages 29
4.4 Object Detection 29
4.4.1 Pre-processing Methods 29
4.4.2 Neural Network Methods 30
4.4.3 Hyperparameters 30
4.4.4 Software Packages 30
4.5 Test on Full-Length Video 31

5 Results & Discussion **32**
5.1 Image Classification 32
5.2 Object Detection 36
5.3 Performance on full-length Video 37

6 Conclusions **43**

Appendices **45**

A Research Code **46**

List of Figures

1.1 A schematic of how cameras are mounted on penguins. Source: Thiebot et al. (2017) 4

2.1 Biological neurons and their networks, (a), formed the basis of artificial neural networks, (b), a machine learning algorithm. 6
2.2 A schematic representation of a perceptron, highlighting the computational process that takes place within a neuron to produce the output $\hat{y}$. 6
2.3 Artificial neural network with two input neurons and a bias term fully connected to a hidden layer with two neurons that terminates with a single neuron in the output layer. 7
2.4 Different activation functions used to perform non-linear transformations on the sum of the product of the input and its associated weights. 8

3.1 Example of the convolution process, where a 3×3 filter, W, is a passed through an 8×8 input image, x, to produce the reduced 6×6 output feature map, F 17
3.2 Example of 2×2 max pooling on a 6×6 feature map, where the highest value in 2×2 regions is selected resulting in a smaller 3×3 matrix. 18
3.3 An example of full CNN architecture. Input image is passed through one or more convolution and pooling layers, then connected to fully connected layers, before finally producing an output, in this case, the probability of being prey or non-prey event 18
3.4 A residual block that forms the basis of the ResNet architecture. The block contains a skip connection that copies the input x to the output, resulting in output mapping being $F(x) + x$. Source: He et al. (2016) 19
3.5 Illustration of the process of the YOLO model. With the problem framed as a regression task, the YOLO model is able to unify the separate components of object detection, namely bounding box predictions (shown in the upper branch) and class predictions (shown in the lower branch), into a single neural network. Source: Redmon et al. (2016) 21
3.6 Illustration of the YOLOv3 architecture that has 53 convolutional layers, consisting of convolutional layers, residual blocks and detection layers. Source: Valdez (2020) 22

3.7 The Precision-Recall Curve. The true precision curve (in blue) has "wiggles" caused by small variations in the ranking of detections. The interpolated precision curve (in red) eliminates this problem. 24
3.8 Subjective ground truth bounding boxes for an image of a jellyfish, with the red box only including the body of the jellyfish, with the large black box also includes the tentacles of jellyfish 24

4.1 Examples of the 1280 × 720 pixel images extracted from penguin borne videos. (a) part of a school of fish visible that is classed as a prey event, while (b) an empty underwater image of the ocean is classed as a non-prey event. 27
4.2 ResNet50 architecture (He et al. 2016) adapted with additional fully connected layer, and output layer with two units. 28

5.1 Plot of prey events predictions (in red),i.e, both jellyfish and schools of fish, by the model compared to the ground truth labels (in blue). Clips a, b and c are different segments from the same unseen video and as such are predicted together, while predictions for clip d were made after the model was retrained to include misclassified images from the previous clips. 34
5.2 Images that the detection model was able to identify correctly. Frames (a) and (b) contain prey events, a jellyfish and a school of fish respectively. Frames (c) and (d) were correctly classed non-prey events. 35
5.3 Images that the detection model was unable to identify correctly. Frame (a) contains a prey event, i.e., a jellyfish, that was classified as a non-prey event. Frame (b) contains air bubbles and debris that was classed as a prey event. Frames (c) and (d) were incorrectly classed prey events when they contain the backs of the penguin carrying the camera. 35
5.4 Images that the isolation model was able to identify correctly. Frames (a) and (b) contain jellyfish, while frames (c) and (d) contain schools of fish respectively. 39
5.5 Images that the isolation model was unable to identify correctly. Frame (a) contains a jellyfish and penguin which the model labeled as a fish. Similarly, in frame (b) the model identified the jellyfish, however, missed the school of fish that was also present. While no detection were made in frames (c) and (d) though the images contains a school of fish and a group of jellyfish respectively. 39
5.6 ResNet-basedmodel's predictions (in red) of prey events on a full-length video compared to the ground truth labels (in blue). The 51 frames containing prey are distributed over 4 distinct prey events (blue vertical bars). 41
5.7 YOLO model's predictions (in red) of prey event bounding boxes on predicted prey events from ResNet-based model, compared to the ground truth labels (in blue).The 51 frames containing prey are distributed over 4 distinct prey events (blue vertical bars). 41
5.8 YOLO model's predictions (in red) of prey event bounding boxes on a full-length video compared to the ground truth labels (in blue). The 51 frames containing prey are distributed over 4 distinct prey events (blue vertical bars). 42

List of Tables

4.1 The number of images in each class that made up the train, validation, and test sets. The images were split by video such that similar images were placed in the same set. 27
4.2 The different hyperparameters tested for the image classification model. . . . 29
4.3 The number of data points for each prey type that made up the train, validation, and test sets. Images may contain more than one object, such that the sum is greater than the total number of images. The images were again split by video such that similar images were placed in the same set. 30
4.4 The different hyperparameters that were tested. With the YOLO default values marked with an * 30

5.1 Summary of the top 5 results from hyperparameter tuning of the ResNet-basedImage Classification model on the validation set, in descending order of accuracy. 33
5.2 Comparison of the performance measures of the best performing ResNet-based models with different additional layers. 33
5.3 The performance measures of the final ResNet-based model on the two additional videos 33
5.4 Summary of performance metrics of final trained model on validation set before and after adding semi-supervised learning process. 34
5.5 Performance metrics of the test set on the final trained model after the inclusion semi-supervised learning process. 36
5.6 Summary of results from hyperparameter tuning of the YOLOv3 model on the validation set, where * denotes the default value of that hyperparameter, and the best performing value is highlighted in bold 38
5.7 Summary of results from hyperparameter tuning of the YOLOv3 model on the test set, where * denotes the default value of that hyperparameter. 38

Chapter 1

Introduction

1.1 Background

Prey availability is one factor that affects the foraging habits of a species. Understanding these behaviours is essential to learning more about ecological behaviour and predicting how variations in environmental conditions can possibly affect ecosystems. It also allows ecologists to refine their perception of the ecological niche of predators and prey, specifically marine life where a lot is still unknown (Thiebot et al. 2017).

Technological advances have created opportunities for ecologists to expand their knowledge of marine life. These technological advances include improved digital photography that collects large numbers of higher resolution images, increased capabilities of computers and software to manipulate, store and analyse images, as well as advances in robotics that lead to the construction of devices such as autonomous underwater vehicles (AUVs) and remotely operated vehicles (ROVs) that are able to capture thousands of images in one deployment (Beijbom et al. 2015). For example, the use of AUV surveys (Lebrato et al. 2012) has aided research into the ecological niche of jellyfish and other gelata, and this has led to evidence that these organisms may play a bigger role in the ocean's carbon cycle than initially thought (Lebrato et al. 2012, Doyle et al. 2014).

Wildlife monitoring and analysis are active research fields in ecology with huge amounts of data being produced from cameras, audio recorders, animal-borne video recorders and many other digital technologies (Verma & Gupta 2018). These devices have led to an influx of raw data that allow ecologists to study the lives of wildlife with minimal disturbance to the animals. Video and audio footage provide researchers with images and acoustic data to study populations, distributions, and behaviours of wildlife.

However, the speed at which data is being created significantly outpaces the speed at which data is being processed. This is because, traditionally, the extraction of information has been done manually either by experts, that is, by researchers and their teams, or by volunteer groups through online crowdsourcing sites (e.g. Amazon Mechanical Turk). This entails people watching, listening, or reading through several hundred hours of footage to extract useful information. This method is both time-consuming and costly, and as a result, leaves

large amounts of the data underutilised (Norouzzadeh et al. 2018).

Automating the information extraction process would speed up the processing of existing datasets, increase the amount of data that could be processed in future, and free up time for researchers to focus on achieving their scientific, management and protection mission (Norouzzadeh et al. 2018).

In the realm of machine learning, there have been many developments in deep learning techniques, specifically in computer vision. Deep learning techniques have been applied to many fields such as medicine (Litjens et al. 2017) and agriculture (Kamilaris & Prenafeta-Boldú 2018). Further research has been looking into their application into ecology, but these methods remain relatively underused, and many applications continue to annotate data manually (McInnes et al. 2017).

Deep learning algorithms have been seen to excel at computer vision tasks (Krizhevsky et al. 2012), such as automatically detecting objects (Norouzzadeh et al. 2018, Girshick 2015), with deep learning models winning the image classification task of the annual ImageNet Large Scale Visual Recognition Challenge [1] consecutively since 2012 (Russakovsky et al. 2015), and as such can be used to identify animals in photographic images. In certain studies, high classification accuracy has been achieved, thus making deep learning models particularly useful for detection and isolation tasks (Christin et al. 2019).

Based on these properties of deep learning algorithms, this dissertation develops a model for identifying the little penguin's two major prey types, small pelagic fish e.g. anchovy, and jellyfish, from images extracted from animal-borne video recordings. The model is divided into two parts, with the input images considered independently, i.e., with no temporal relationship from the video. The first sub-model is an image classification model that determines whether any prey is present within a frame. This is then fed into an object detection model that identifies exactly where within the frame the prey event is and labels the type of prey. A prey event is defined as when a known prey of the penguin is seen within a frame and a non-prey event may be an empty image or one containing debris or other penguins, but no prey.

1.2 Research Objectives

The research objectives of this dissertation are as follows:

1. To develop an image classification model based on existing deep learning architectures that would be able to distinguish between prey and non-prey events;

2. To develop an object detection model based on the YOLO framework that draws bounding boxes around the prey and identifies the type of prey in the images; and

[1] The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) allows researchers to evaluate algorithms for object detection and image classification on a large scale, with a large annotated dataset of a variety of objects

3. To assess the accuracy of the models developed, and any features affecting this accuracy, and whether the developed models are sufficiently accurate to assist manual processing.

1.3 Little Penguins

Little penguins (*Eudyptula minor*) also known as fairy penguins, are found along the southern coasts of Australia, some islands off the southern Australian coast as well as in New Zealand (Burns 2006). Penguin Island marks the northern and western limits of the little penguin breeding range (Burns 2006). Little penguins are not listed as an endangered species; however, studies, such as that done by Dann (1992), have shown that their populations are under threat with declining numbers especially in areas where they are situated close to growing human settlement and activities (Preston et al. 2008).

Understanding the feeding behaviour of animals is a critical part of life history evolution. The rate of energy acquisition through feeding determines how energy is used in processes such as growth or reproduction. Seabirds, such as little penguins, are central place foragers, meaning that they reproduce on land and gather food at sea (Ropert-Coudert et al. 2006).

Little penguins are visual hunters, limiting their foraging time to daylights hours (Cannell & Cullen 1998). Relatively little is known about their foraging efforts, but studies based in Western Australia have shown that during breeding, the penguins almost exclusively stay within a fifteen to twenty kilometre radius of the colony, where the waters are shallow (Ropert-Coudert et al. 2006).

The feeding of little penguins is also limited by their hunting and prey processing abilities as well as prey encounter rate (Ropert-Coudert et al. 2006). These constraints make seabirds, like little penguins, interesting study animals for behavioural ecology, especially with developments in recording and tracker equipment, that can be used to monitor and understand their activities at sea (Ropert-Coudert et al. 2006). For example, little penguins can only gain resources within a short distance of their colony in order to return regularly to feed their chicks. This behaviour in turn allows researchers a central location to attach and remove data logger equipment used to study the animals (Preston et al. 2008).

Little penguins mainly eat small schooling fish such as sandy sprat (*Hyperlophus vittatus*), pilchard (*Sardinops sagax*), garfish (*Hyporhamphus melanochir*), blue sprat (*Spratelloides robustus*) and anchovy (*Engraulis australis*). They have also been documented to eat squid (*Idiosepius notoides*) and jellyfish (Ropert-Coudert et al. 2006). Though penguins have been documented to consume jellyfish, due to their low nutritional content and rapid digestive deterioration, they may often be overlooked in dietary studies (Sutton et al. 2015). A study by Flemming et al. (2013) shown that little penguins eat a diverse range of species and may switch between species, mainly in response to temporal variation in prey availability. For example, little penguins in New Zealand ate higher proportions of lower quality cephalopods than those in Australia (Flemming et al. 2013). Changes in food quality and availability can affect seabirds' population, and as top predators in marine ecosystems, changes in their diets may indicate changes happening in the inshore marine ecosystems (Flemming et al. 2013).

Little penguins are quite flexible in whether they forage alone or with a group (Camprasse et al. 2017). The study by Sutton et al. (2015) showed that little penguins have a higher probability of associating with conspecifics, i.e. group foraging, when hunting prey that were aggregated than when prey was solitary, and that prey capture rates were higher when the penguins hunted schooling prey rather than solitary prey. Group foraging allows the predators to over-power large prey or aggregating small prey, for better capture rates (Sutton et al. 2015).

All this information about the behaviour of penguins has been made available through advances and increased accessibility of tracking technology, such as the video logger in Figure 1.1. This technology allows ecologists and biologists to better monitor the different behaviours of animals in the wild. For example, ecologists now know that at least four species of penguins including the little penguin eat gelata organisms such as jellyfish. However, with several hundred hours of video recordings, researchers are able to see that penguins were actively pursuing and eating jellyfish (Thiebot et al. 2017).

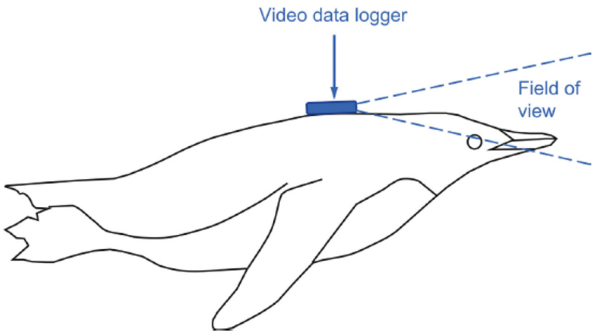


Figure 1.1: A schematic of how cameras are mounted on penguins. Source: Thiebot et al. (2017)

1.4 Outline of the Dissertation

This dissertation is comprised of six chapters, including this introductory chapter. Chapter 2 describes the theory of neural networks, while Chapter 3 delves into computer vision applications of convolutional neural networks (CNNs), and their further application in ecological projects. Chapter 4 describes the raw data used, how useful inputs were extracted from it and provides details of the image classification and object detection models. Chapter 5 documents the results of both models and discusses the practical implications of the results. Chapter 6 concludes the research and explores avenues for possible future research.

Chapter 2

Basics of Neural Networks

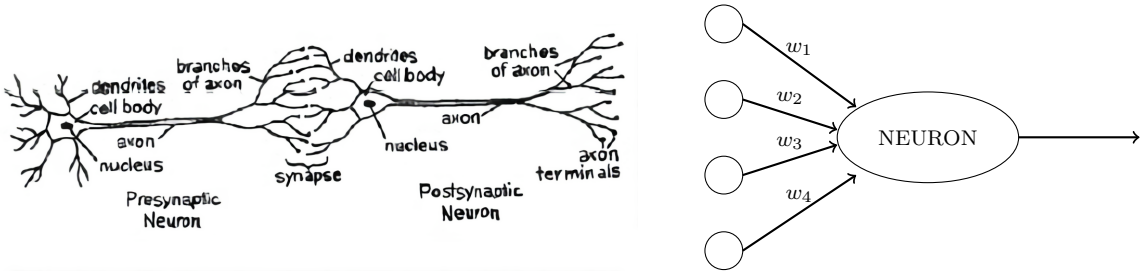
2.1 Artificial Intelligence and Machine Learning

Artificial Intelligence (AI), a term first coined by John McCarthy in 1956 (McCarthy et al. 2006), was defined as "the science and engineering of creating intelligent machines" (McCarthy 2007) that can achieve goals previously reserved for humans, and are able to improve themselves (McCarthy et al. 2006). It is more concisely defined as an effort to automate intellectual tasks normally performed by humans (Chollet 2018). Examples of such tasks include identifying objects either visually or audially, and using human languages. AI is considered the umbrella term that encompasses techniques like machine learning, deep learning, reinforcement learning and many more approaches.

Machine learning, a branch of artificial intelligence, is the study of computer algorithms that are able to learn from data, identify patterns and make decisions with minimal human intervention (James et al. 2013). Machine learning systems are trained rather than explicitly programmed (Chollet 2018). Machine learning is further split into more branches, specifically, supervised learning, unsupervised learning and reinforcement learning.

Supervised learning involves building statistical models for prediction or estimation of an output based on a series of inputs (James et al. 2013). These models are given a training set that contains the inputs and their respective outputs, known as labels, which the model will use to understand the data, and then predict outputs of new input values. On the other hand, unsupervised learning techniques do not have training outputs. The models are used to learn relationships and find structures existing in the dataset (James et al. 2013).

A commonly used machine learning method is artificial neural networks (ANNs), which are loosely based on the human neural network (Figure 2.1). They come from the idea that "a neuron's computation involves a weighted sum of the input values. These weighted sums correspond to the value scaling performed by the synapses and the combining of those values in the neuron" (Sze et al. 2017). A functional operation, typically a simple non-linear transformation, is performed on the combined inputs within the neuron and an output is only generated if the output of this function is above a certain threshold (Sze et al. 2017).



(a) Biological Neuron. Source: Aggarwal (2018)

(b) Artificial Neuron, also known as a perceptron

Figure 2.1: Biological neurons and their networks, (a), formed the basis of artificial neural networks, (b), a machine learning algorithm.

In their original formulation, ANNs were composed of a single "hidden" layer – a single iteration of combining inputs and transforming these into an output. "Deep" neural networks are those with more than one hidden layer – each layer produces possibly multiple outputs, which become the inputs to the next layer. Deep learning (LeCun et al. 2015) will be discussed more thoroughly in Section 2.8.

2.2 Feed Forward Neural Networks

A neural network's basic unit is a perceptron illustrated in Figure 2.2 and its simplest form is a feed-forward neural network (Figure 2.3) which is named such because it does not have feedback connections. The neural network performs a function on the inputs, x, using parametric weights to output an approximation of y, the target variable (Goodfellow et al. 2016). Learning occurs when the approximation, $\hat{y}$, is compared to the target, y, and the weights, w, are adjusted to improve the prediction. This is done through a process known as back-propagation (see Section 2.5.1).

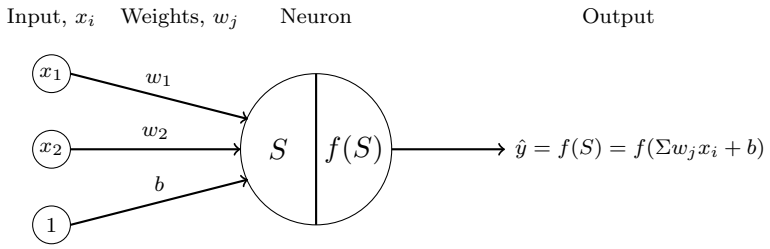


Figure 2.2: A schematic representation of a perceptron, highlighting the computational process that takes place within a neuron to produce the output $\hat{y}$.

The summation, S, of all weighted inputs, $w_j x_i$, is then passed through a nonlinear activation function f, to transform the preactivation level of the neuron to an output, $\hat{y}$ (Vieira et al. 2017). A bias term, b, is also included. The bias accounts for the invariant part of a prediction, and is incorporated as a weight value on a neuron that always has a value of 1 (Aggarwal

2018) (see Equation 2.1).

$$\hat{y} = f\left(\sum w_j x_i + b\right) \tag{2.1}$$

Figure 2.3 is used to illustrate the functionality of a feed-forward neural network. The network has three layers, namely an input layer, followed by a hidden layer, then lastly the output layer. The input layer has a fixed number of nodes, D, that is equal to the number of variables present in the data. These nodes are fully connected to the nodes in the hidden layer. Fully connected or dense layers are layers where all its nodes are connected to all nodes of the next layer. Each connection has an associated weight, w. The number of nodes in the hidden layer, as well as the number of hidden layers, is user-defined. If the neural network has only one hidden layer, it is known as a shallow neural network, while if it has more hidden layers, it is a deep neural network.

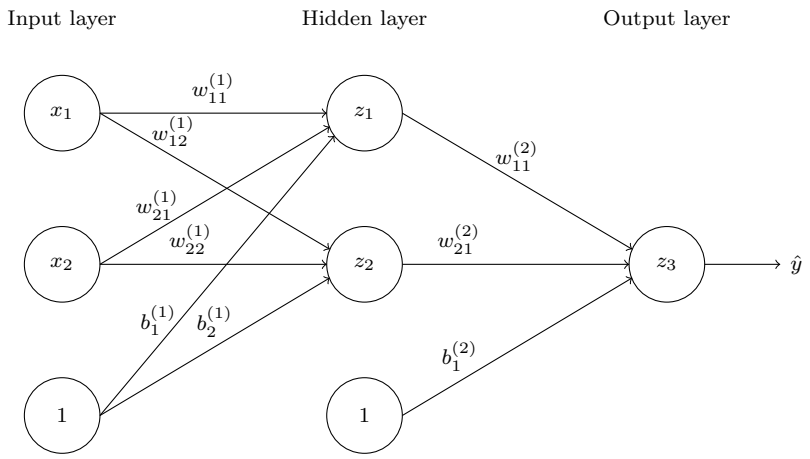


Figure 2.3: Artificial neural network with two input neurons and a bias term fully connected to a hidden layer with two neurons that terminates with a single neuron in the output layer.

$$\begin{aligned} S_j^{(l)} &= \sum_{i=i}^{d^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)} \\ z_j^{(l)} &= f\left(S_j^{(l)}\right) \end{aligned} \tag{2.2}$$

Where:

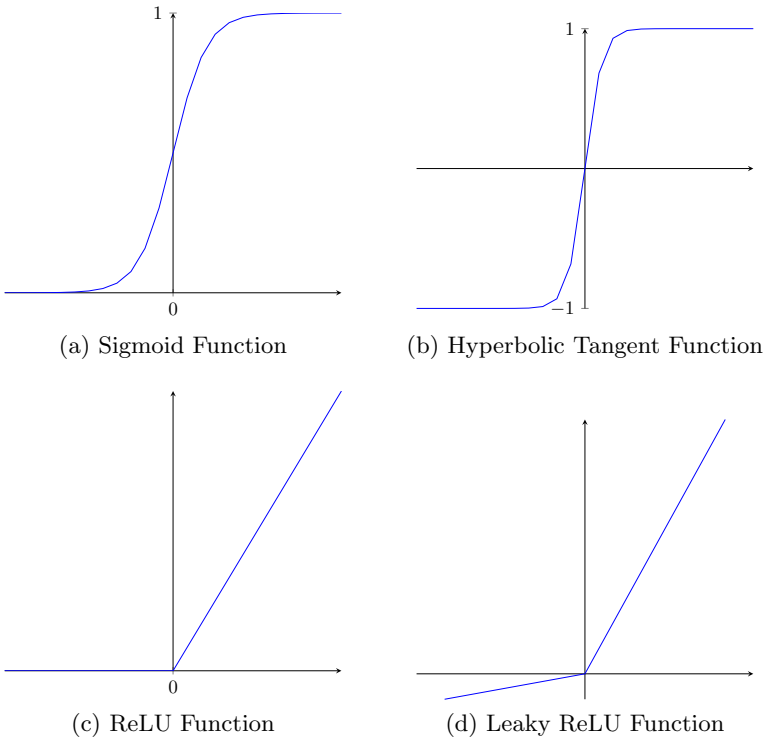
- l is the layer index
- $d^{(l-1)}$ denotes the number of nodes in the previous, $(l-1)$th, layer
- $w_{ij}^{(l)}$ is the weight for the connection from the ith neuron in the $(l-1)$th layer to the jth neuron in the lth layer
- $x_i^{(l-1)}$ is the value of the input from the ith node of the $(l-1)$th layer

- $b_j^{(l)}$ denotes the bias term for the jth node in the lth layer
- $S_j^{(l)}$ is the sum of the weighted input, that is input to the jth neuron in the lth layer
- $f(.)$ denotes the activation function
- $z_j^{(l)}$ is the output of the jth neuron in the lth layer that serves as the input for neurons in the $(l+1)$ layer

To train a neural network, activation functions, a loss function and an optimizer also have to be decided upon (Chollet 2018). All of these concepts are discussed more fully in the sections below.

2.3 Activation Functions

Activation functions perform a non-linear transformation on the summation of the product of the input and its associated weights to produce an output (Vieira et al. 2017). This allows a greater range of relationships to be modelled (Chollet 2018). Activation functions are a critical part of neural network design (Aggarwal 2018). Popular activation functions include sigmoid, Hyperbolic Tangent (tTanh), ReLU (rectified linear unit) (Glorot et al. 2011), and leaky ReLU (Maas et al. 2013). These functions are shown in Figure 2.4.



(a) Sigmoid Function (b) Hyperbolic Tangent Function

(c) ReLU Function (d) Leaky ReLU Function

Figure 2.4: Different activation functions used to perform non-linear transformations on the sum of the product of the input and its associated weights.

A sigmoid function (Equation 2.3) is useful in a situation where one is attempting to predict the probability of a binary class (Aggarwal 2018). The range of possible values is between 0 and 1. It "squashes" arbitrary values into the [0, 1] interval (Chollet 2018).The Tanh activation function (Equation 2.4) is similar to the sigmoid function, however, it has a wider range, as values can be between -1 and +1.

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

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

A ReLU function (Equation 2.5) is the most widely used activation function due to its speed and effectiveness (Chollet 2018). ReLU sends all negative values to zero, and is linear for positive values (Glorot et al. 2011). Deep networks with ReLU units are more easily optimized than networks with sigmoid or tanh units because there is no gradient vanishing effect (Glorot et al. 2011). Vanishing gradients cause slow optimization convergence (Maas et al. 2013). Other variations of ReLU have been proposed such as Leaky ReLU (Equation 2.6) (Maas et al. 2013) and ELU (exponential linear unit) (Equation 2.7) (Clevert et al. 2016). These variations aimed to alleviate problems caused by hard zero activation for non-active units in ReLU by having a small value for negative inputs, achieved by multiplying x by a small number, α. ELU is faster and ensures noise-robust deactivation states unlike Leaky ReLU (Clevert et al. 2016).

$$f(x) = \max(0, x) \tag{2.5}$$

$$f(x) = \max(\alpha x, x) \tag{2.6}$$

$$f(x) = \max(\alpha(e^{x} - 1), x) \tag{2.7}$$

Softmax, Equation 2.8, is another activation function. It is commonly used on the output layer of a classification model that has two or more output classes. It is used to represent probability distributions over discrete variables over K different classes (Goodfellow et al. 2016). The output of each unit is a value between 0 and 1 for which the sum of the outputs adds up to 1.

$$\text{softmax}(x_n) = \frac{e^{x_n}}{\sum_{i=1}^{K} e^{x_i}} \tag{2.8}$$

2.4 Loss Functions

Loss functions, also known as objective functions, are used to measure the performance of the network. The loss function measures the difference between the predicted output produced by the network and the true output value. The loss function provides a precise mathematical way to evaluate the quality of a network's predictions over any set of observations.
This is used as a mechanism for:

a) deciding how to adjust weights in the backpropagation algorithm (iteratively reducing the sum of loss function values across a set of "training" observations using gradient descent, as described in Section 2.5.1);

b) choosing between different neural network architectures and hyperparameters (by selecting the one that gives the lowest sum of loss function values across a set of "validation" observations); and

c) providing a final assessment of a model's ability to generalize to new data (by directly calculating the sum of loss function values across a set of "test" observations).

Commonly used objective functions include squared error loss and squared absolute error for regression problems, and cross-entropy loss for classification problems.

2.4.1 Cross Entropy Loss Function

Cross entropy loss is Bernoulli negative log-likelihood function, that is used on classification problems, i.e., when the output label, y, is discrete (Goodfellow et al. 2016). For classification problems, such as the one addressed in this dissertation, suitable functions are binary cross-entropy or categorical cross-entropy.

In the case that y is binary, $y \in \{0, 1\}$, binary cross-entropy is applicable (Equation 2.9). A sigmoid activation function can be used in the final layer to ensure $\hat{y}$ lies between 0 and 1. This choice of activation function also dictates the number of output neurons, with sigmoid output activation requiring one neuron.

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln \hat{y}_i + (1 - y_i) \ln(1 - \hat{y}_i) \tag{2.9}$$

For classifications problems where the number of classes,K, is greater than two, categorical cross-entropy (Equation 2.10) is more appropriate. Softmax activation is a suitable function for the output layer and will require the output layer to have K neurons. Softmax ensures $\hat{y_k}$ values that are between 0 and 1.

$$L(\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \ln \hat{y_{ik}} \tag{2.10}$$

2.5 Optimization

2.5.1 Backpropagation

In a neural network, appropriate weights are produced by iteratively changing the weights after each forward pass to decrease the total loss or error. The challenge was that the total loss value recorded would be a composite of errors accumulated by the entire network. Backpropagation is an algorithm that is based on the application of gradient descent and chain rule for differentiation (LeCun et al. 1988). It is used to estimate the weight of a neural network by starting with the final loss value and working backwards from the output layer

to the first hidden layer by applying chain rule to compute the contribution of each weight to the loss value (Chollet 2018).

The derivative of the loss function with respect to the weights can be expressed as:

$$\frac{\partial E}{\partial w_{ji}^{(l)}} = \frac{\partial E}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} \times x_i^{(l-1)} \text{ for all } i, j, l \tag{2.11}$$

Where δ_j^l is the error in the j^{th} neuron in the l^{th} layer. Backpropagation provides a way to compute the errors for every neuron of the model, which are in turn used to compute the gradients of the loss function, E, with respect to the weights by using equation 2.11.

In the output layer, the error equation can be expressed as

$$\begin{aligned} \delta_j^{(L)} = \frac{\partial E}{\partial s_j^{(L)}} &= \frac{\partial E}{\partial z_j^{(L)}} \times \frac{\partial z_j^{(L)}}{\partial s_j^{(L)}} \\ &= \frac{\partial E}{\partial z_j^{(L)}} \times f'(s_j^{(L)}) \end{aligned} \tag{2.12}$$

Where $\frac{\partial E}{\partial z_j^{(L)}}$ and $f'(s_j^{(L)})$ can both be computed easily given the form of the loss and activation functions in the output layer.

For the hidden layers, the error can be defined backwards as:

$$\begin{aligned} \delta_i^{(l-1)} = \frac{\partial E}{\partial s_i^{(l-1)}} &= \frac{\partial z_i^{(l-1)}}{\partial s_i^{(l-1)}} \times \frac{\partial E}{\partial z_i^{(l-1)}} \\ &= f'(s_i^{(l-1)}) \sum_{j=0}^{d^{(l)}} \frac{\partial E}{\partial s_j^{(l)}} \times \frac{\partial s_j^{(l)}}{\partial z_j^{(l-1)}} \\ &= f'(s_i^{(l-1)}) \sum_{j=0}^{d^{(l)}} \delta_j^{(l)} \times w_{ij}^{(l)} \end{aligned} \tag{2.13}$$

2.5.2 Optimization Functions

Gradient descent is an iterative numerical optimization algorithm that is used to minimize the loss function E by adjusting the weights in the opposite direction of the gradient of the loss function with respect to the weights, w (Ruder 2016). The learning rate η determines the size of the steps we take to reach a local minimum.

$$w_{updated} = w_{old} - \eta \nabla E \tag{2.14}$$

where η is the learning rate and E is the loss or cost function.

Variants of gradient descent exist such as the commonly used Stochastic Gradient Descent (SGD) which performs weight updates using subsets of the dataset. This eliminates redundant computations, that is, recomputing the gradients for similar examples, therefore it is able to converge faster than gradient descent (Ruder 2016). Other optimisation functions used in deep learning algorithms include Adaptive Moment Estimation (Adam) (Kingma & Ba 2014), RMSProp (Tieleman et al. 2012) and AdaGrad (Duchi et al. 2011). The Adam optimizer is a commonly used optimization function (Schneider et al. 2018), as it utilises the advantages of earlier optimizers, RMSProp and AdaGrad, to outperform them (Kingma & Ba 2014) (Ruder 2016). For this reason, it was used in this investigation.

2.6 Generalization

Generalization refers to how well a trained model performs on a related dataset that it has not seen before (Chollet 2018). The training and validation data are used to train and tune the model to ensure that it is optimised for the dataset. However, when a model is too specific to the training data, it may perform poorly on unseen data, i.e. the test data. This is known as overfitting. Regularization refers to any modification made to a model that is used to combat the effects of overfitting, to ensure a more generalised model without affecting the training error (Goodfellow et al. 2016).

2.6.1 Weight Regularization

A model may learn a very complex function that fits the training data almost perfectly but performs poorly on unseen data. One way to mitigate this is to penalise the complexity of the model by forcing their weights to take small values, this is known as weight regularization (Chollet 2018). This is done by adding a penalty for having large weights to the loss function. The penalty may be in one of the following two forms:

1. L1 (Lasso) regularization: The cost added is proportional to the absolute value of the weights coefficients.
$$E_{reg} = E + \lambda \sum_{i=1}^{D} |w_i| \tag{2.15}$$

2. L2 (Ridge) regularization: The cost added is proportional to the square of value of the weights coefficients.
$$E_{reg} = E + \lambda \sum_{i=1}^{D} w_i^2 \tag{2.16}$$

where D is the number of nodes, and λ is related to a user-defined threshold, τ, that constrains the parameters of the model closer to zero (James et al. 2013).

2.6.2 Early Stopping

When a model is trained for a long time such that overfitting occurs, it it also observed that while the training error continuously decreases, the validation error starts to rise again over time. Early stopping is a strategy that stores the model weights at a low validation error,

and updates every time the validation errors improves, i.e. decreases. The training may also be terminated early if no improvement is seen after a set number of epochs. Once training is terminated, the model weights from the best validation performance are selected as the final weight values (Goodfellow et al. 2016).

2.6.3 Dropout

Dropout (Srivastava et al. 2014) reduces overfitting by preventing complex co-adaptations. This is achieved by setting the weights of neurons to zero with a certain probability, during the forward pass, and the gradients of these neurons also being zero on the backwards pass. This ensures that the models learn several independent representations of the same data, and thus improves the generalization of the model (Goodfellow et al. 2016).

2.7 Hyperparameter Optimization

The hyperparameters are the parameters whose values are selected by the user to define the model architecture. These values are not adapted by the algorithm itself during the learning process (Goodfellow et al. 2016) but are tuned to find the optimum combination that achieves the best performance of the model. The optimum combination is not immediately known, and thus several combinations must to be run to explore the impact of each hyperparameter.

This can be done using different search methods such as grid search or random search. The grid search is an exhaustive method that searches all combinations of the hyperparameter values provided and selects the model which results in the best performance. Random search uses values that are randomly sampled from a provided statistical distribution of each hyperparameter (Bergstra & Bengio 2012).

The purpose of the validation set is to optimize a model's architecture and as such is used to determine which combination of hyperparameters performed optimally. Common hyperparameters include learning rate, dropout rate, the number of layers used, the number of units or filters in each layer, and activation function Chollet (2018).

2.8 Deep Learning and Transfer Learning

Deep learning is an area of study in which the neural networks are able to learn higher-level features as functions of lower level features learned from data (McCann & Reesman 2013). Deep neural networks are larger than traditional neural networks in that their depth of the network is increased by adding more hidden layers. For a neural network to be considered deep, it must have more than three layers, that is, more than one hidden layer. Typically, the number of layers used in deep learning range from more than five and up to one thousand (Sze et al. 2017).

Deep Neural Networks are capable of learning high-level features with more complexity and abstraction than shallow networks. Shallow neural networks tend to preform better on smaller

datasets (Schindler et al. 2016). Previously, many argued that increasing the number of hidden layers did not improve the accuracy of neural networks (Csáji et al. 2001). They justified this assertion with the Universal Approximation Theorem, which claims that a standard multilayered perceptron with a single hidden layer and a finite number of hidden neurons can approximate any function when given the appropriate weights (Csáji et al. 2001). However, studies such as that done by Mhaskar et al. (2016) proved that deep networks can approximate functions with the same accuracy as shallow neural networks but with exponentially lower number of training parameters. With developments in computational power and access to more data, deep neural networks proved to be able to train effectively and produce better results (Schindler et al. 2016).

These developments led to more exploration of many different deep architectures. These include deep feed-forward, convolutional neural network (CNN), recurrent neural network (RNN) (Medsker & Jain 2001), Long/Short Term Memory (LSTM) (Hochreiter & Schmidhuber 1997) to name a few. Convolutional neural networks will be discussed in greater detail in Section 3.1.

Another development related to deep learning is transfer learning. Transfer learning is a technique where a model that was previously trained on a previous dataset that has similar characteristics, is now used on a new, often smaller dataset to capitalise on the knowledge gained from the previous training exercise (Pan & Yang 2009).

A model that was previously trained to address a specific problem, X, would ideally provide better initial weights for a new problem, Y, than a model with randomly initialised weights, provided that the two problems are related. Thus, it provides a better starting point and would take a shorter time to reach optimal values for problem Y, since learned filters such as edge or colour detectors, can be used without having to be re-learned on new data. This means that transfer learning helps reduce the required size of the training dataset for problem Y since several filters have already been established from problem X (Pan & Yang 2009). Thus transfer learning is especially useful in cases where large amounts of training data would be expensive or difficult to obtain (Weiss et al. 2016). Typically, the possible output values for the pre-trained network are not the same as those of the new task. For example, a task identifying a single species may make use of a pre-trained network built from a dataset of hundreds of species, or even on a dataset of non-animal image. This requires removing the final layer of the pre-trained network, replacing it with a final layer appropriate to the new task, and estimating at least the weights of the final layer. Weights of other hidden layers can also be estimated, starting from the pre-trained values, in a process known as "fine-tuning" (Pan & Yang 2009).

It is possible to keep the weights from the pre-trained network in some of the hidden layers, this is known as freezing a layer. The weights in a layer that has been frozen cannot be modified further and are called non-trainable weights/parameters. Trainable weights or parameters are the weights in all the other layers that are adapted through the learning process. Freezing layers is useful to retain some of the representations learned from the pre-trained network (Chollet 2018). Also, reducing the number of trainable parameters reduces

the computational time of training, while not compromising too much on accuracy (Sagar 2019).

Chapter 3

Computer Vision

The idea of computer vision first started in the early 1970s. It is the field of study in AI dedicated to training computers how to see and understand the meaning in digital images and videos (Brownlee 2019). Originally, it was believed that solving the "visual input" problem would be an easy step along the path to solving more difficult problems such as higher-level reasoning and planning, however, it proved to be more difficult than anticipated (Szeliski 2011). It was not until AlexNet (Krizhevsky et al. 2012), a deep neural network, won the ImageNet Classification (Everingham et al. 2010) contest in 2012 that huge strides were made in this field. AlexNet reduced the error rate by almost 10%, this sparked a trend of addressing computer vision problems using deep learning approaches together with graphics processing units (GPUs), specialised electronic circuits designed to efficiently manipulate computer graphics and image processing (Sze et al. 2017).

3.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have become increasingly popular for applications in image recognition, classification, and detection (LeCun et al. 1998). CNNs were inspired by the visual cortex of the brain that has alternating layers of simple and complex cells (Hubel & Wiesel 1962). Analogously, a CNN has convolutional and pooling layers (Rawat & Wang 2017).

The convolutional layer acts as feature extraction layer that learns the important features from its input image, storing the results in a matrix, known as a feature map (Rawat & Wang 2017). This was inspired by the regions of cells in the visual cortex would be activated when specific regions in the visual field are excited which also depend on the shape and orientation of the objects in the visual field (Aggarwal 2018).

A convolutional layer gets its name from the mathematical operation convolution that is performed within this layer. In this layer, a convolution (Equation 3.1) is a dot-product operation that is performed between a grid-structured inputs, x and similar grid-structured set of weights, W that produces a third grid-structure that expresses key features of the input

in relation to the filter (Aggarwal 2018).

$$\mathrm{F} = x * W \tag{3.1}$$

where F is the output feature map, x is an input image, $*$ is the convolution operator, and W is a filter, also known as a kernel. The filter is a matrix of trainable weights, a parameter that is adjusted through the learning process. Several filters applied to an input image or feature map is referred to as a filter bank. The convolution process is repeated, applying different filters that result in several feature maps that represent different characteristics or patterns found in the input image, such as colour clusters, or the presence or absence of lines (Schneider et al. 2018, Yamashita et al. 2018). The convolution is also passed through a non-linear activation function, usually ReLU.

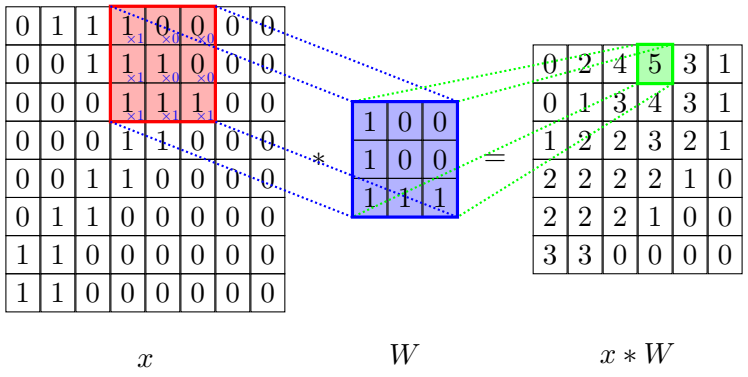


Figure 3.1: Example of the convolution process, where a 3 × 3 filter, W, is a passed through an 8 × 8 input image, x, to produce the reduced 6 × 6 output feature map, F

The grid structured sets on which the dot-product is performed are known as tensors, and they are essential to CNNs. Tensors are multidimensional arrays, that is, a generalization of vectors and matrices to an arbitrary number of dimensions (Chollet 2018). For example, a sequence of words can be represented as a vector, i.e., an order 1 tensor; a grayscale image can be represented by its pixels in a matrix form, i.e., as an order 2 tensor; while a colour image would be an order 3 tensor, with the colour channels creating the third dimension (Wu 2017). CNNs were designed to work on inputs that have strong spatial dependencies in local regions of the grid structure, as such they perform well in natural language and computer visions tasks, where the inputs are structured sentences and images (Aggarwal 2018).

A subsampling layer, known as a pooling layer, is another key part of CNNs, which usually comes after a convolution layer. The aim of this layer is to reduce the computational requirements and as well as increase robustness by dividing the feature map into regions and only returning a single value from each region (Schneider et al. 2018). Common pooling techniques include average pooling that would average the values in a region; and max pooling which returns the highest value in the region instead. Max pooling method is illustrated in Figure 3.2.

After a sequence of alternating convolutional and pooling layers, the network is connected to traditional fully connected layers, in which every node in the input layer is connected to every node in the output layer, that then lead to an output, as shown in Figure 3.3. Weight

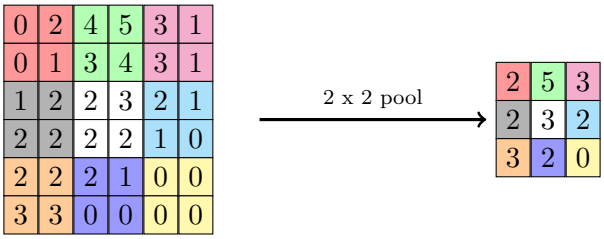


Figure 3.2: Example of 2×2 max pooling on a 6×6 feature map, where the highest value in 2×2 regions is selected resulting in a smaller 3×3 matrix.

parameters are estimated using gradient descent with backpropagation, as in the case of feed-forward ANNs.

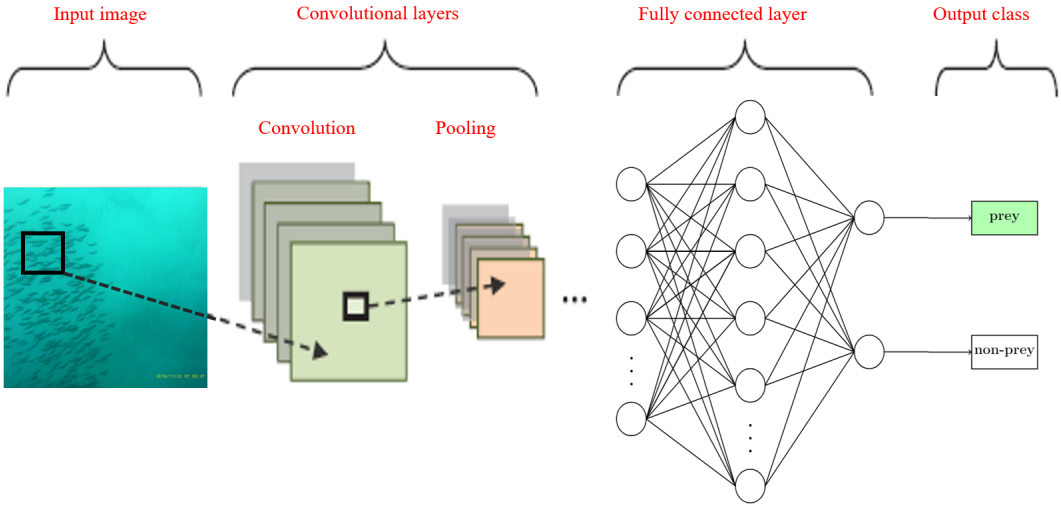


Figure 3.3: An example of full CNN architecture. Input image is passed through one or more convolution and pooling layers, then connected to fully connected layers, before finally producing an output, in this case, the probability of being prey or non-prey event

,

3.2 Image Classification

Image classification is one of the fundamental tasks in computer vision. It involves interpreting the scene present in an image and assigning it to a predefined class (Rawat & Wang 2017). Deep learning models, i.e. models with distinct architectures and fitted weights, that have been used for image classification include AlexNet (Krizhevsky et al. 2012), VGG (Simonyan & Zisserman 2014), GoogleNet (Szegedy et al. 2015), and ResNet (He et al. 2016). These models differ primarily in their architectures and in incremental technical improvements, often in the way that the backpropagation algorithm is applied.

Our work uses the ResNet image classification model, a deep residual learning framework used for image classification. It was developed by He et al. (2016) and won the 2015 ImageNet competition.

A problem faced by many deep neural networks is that of vanishing or exploding gradients which is caused by the increased depth of networks. This, in turn, hampers the ability of the model's learning process to converge (Aggarwal 2018). It was seen that as the depth of a network increases, the accuracy gets saturated and then degrades rapidly. The deep residual learning framework was developed to address the degradation of training accuracy (He et al. 2016).

Instead of representing the desired underlying mapping, $H(x)$, solely by several stacked nonlinear layers $F(x)$, ResNet fits the desired mapping to $F(x) + x$. This is based on the hypothesis that it is easier to optimise the residual mapping, x, than to optimize the original, unreferenced mapping (He et al. 2016). Figure 3.4 shows the residue block and skip connection of the ResNet architecture.

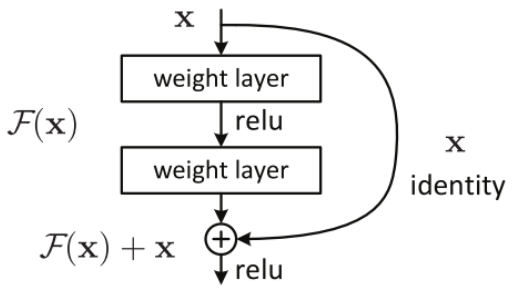


Figure 3.4: A residual block that forms the basis of the ResNet architecture. The block contains a skip connection that copies the input x to the output, resulting in output mapping being $F(x) + x$. Source: He et al. (2016)

Shortcut or skip connections are those that skip one or more layers and enable the copying of input values at layer i and adding of it to the output of the layer $(i + r)$ for $r > 1$. The shortcuts do not increase the number of parameters or add computational complexity. Thus the network is still trained with backpropagation, as a sort of "superhighway" is created through the shortcut connections (He et al. 2016, Aggarwal 2018).

3.2.1 Model Evaluation

When assessing the performance of a classifier the following metrics were used. The simplest of which is the accuracy of the model. Accuracy (Equation 3.2) is number of observations in the test set that is predicted correctly over all observations. Another measure is sensitivity (also known as recall) (Equation 3.3), which is the percentage of correctly predicted positive observations over the total number of positive observations (Zachariah et al. 2014). Similarly, specificity (Equation 3.4) is the percentage of correctly predicted negative observations over the total number of negative observations (Zachariah et al. 2014). Precision (also known as

positive predictive value) (Equation 3.5) is the percentage of correct positive prediction over all positive predictions (Parker 2013).

A perfect classifier would be 100% sensitive and 100% specific, which corresponds to perfect precision and recall. However, this is difficult to achieve, since models that have higher precision tend to have lower recall. F1-score (Equation 3.6) is a measure that calculates the balanced harmonic mean of precision and recall. It assumes that precision and recall are of equal importance. This assumption has caused the F1-score to receive criticism as in some cases, one may be significantly more important, and thus the performance would differ significantly too (Parker 2013).

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

$$\text{sensitivity (recall)} = \frac{TP}{TP + FN} \tag{3.3}$$

$$\text{specificity} = \frac{TN}{TN + FP} \tag{3.4}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{3.5}$$

$$\text{F1-score} = \frac{2 \times precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{3.6}$$

where TP, TN, FP, FN are true positive, true negative, false positive and false negative respectively.

Since, the training, validation and test sets had balanced classes, it was decided to use accuracy as the deciding metric for the selection of the final model. Accuracy is a commonly used and simple metric (Wang et al. 2007) that is considered reasonable when the classes are balanced which the sets are. When the classes are imbalanced, the choice of metric becomes more important, and metrics like the F1-score are preferable (Chicco & Jurman 2020).

3.3 Object Detection

Another task in the field of computer vision is object detection. Object detection involves estimating the object class and the location of objects within the images (Szegedy et al. 2013). Prior to recent developments, object detection problems were considered challenging to model due to significant variations in colour and texture, complexity of the background behind objects, and characteristics such as lighting, and size of the object (Papageorgiou et al. 1998).

Many object detection methods are built on top of classifier systems, that is, classification models, such as, deformable parts models. Deformable parts models (DPM) use a sliding

window to scan through evenly spaced locations over the entire image (Redmon et al. 2016), at each step evaluating whether the window covers the target objects. This approach makes the process very expensive and difficult to scale (Erhan et al. 2014). More modern object detection models that use deep learning include R-CNNs (Region-based Convolutional neural networks) (Girshick et al. 2014), its successors Fast R-CNN (Girshick 2015) and Faster R-CNN (Ren et al. 2017), as well as YOLO (You Only Look Once)(Redmon et al. 2016). Our work makes use of YOLO, which we discuss in more detail below.

YOLO frames the problem as a regression problem rather than a re-purposed classifier. This means a simple pipeline of a single convolutional network is created that results in the prediction of the vertices of multiple bounding boxes and the class probabilities for the boxes, as illustrated in Figure 3.5. They refer to the model as unified detection because it unifies the separate components of object detection, namely bounding box predictions and class predictions, into a single neural network (Redmon et al. 2016).

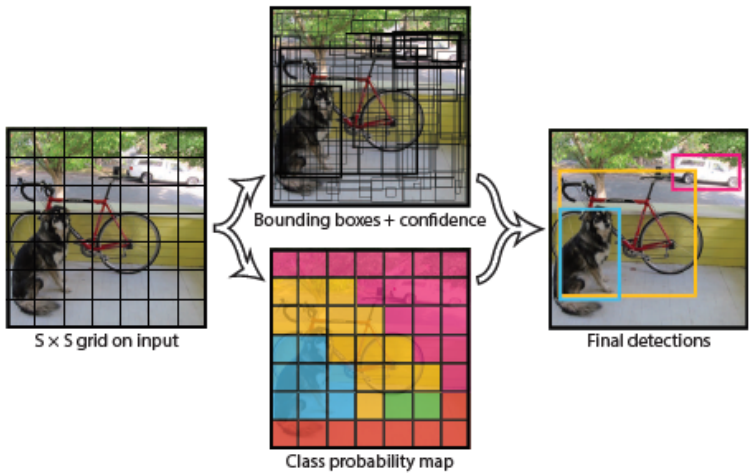


Figure 3.5: Illustration of the process of the YOLO model. With the problem framed as a regression task, the YOLO model is able to unify the separate components of object detection, namely bounding box predictions (shown in the upper branch) and class predictions (shown in the lower branch), into a single neural network. Source: Redmon et al. (2016)

YOLO divides the input image into an $S \times S$ grid. Each grid cell is responsible for predicting an object if the centre of that object falls within that cell. Each cell predicts B bounding boxes and confidence scores for the boxes. The confidence scores show how confident the model is that the box contains an object, i.e., the probabilty of the object being present, and how accurately it thinks it predicted the box, i.e. the intersection over union (IoU) of the ground truth and the predicted box (Redmon et al. 2016). This is shown in Equation 3.7.

$$\text{confidence} = \Pr(Object) \times \text{IoU}_{pred}^{truth} \tag{3.7}$$

where IoU is the ratio of the area of intersection and the area of union of the predicted and ground truth bounding boxes.

Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i \mid \text{Object})$, i.e., the probability of an object being in class i, given that the grid cell contains an object. The conditional class probabilities are then multiplied with the individual box confidence predictions. This gives class-specific confidence scores for each box (Equation 3.8). These scores account for both the probability that the object in the box is of that specific class and how well the predicted box fits the object (Redmon et al. 2016).

$$\Pr(Class_i \mid Object) \times \Pr(Object) \times \text{IoU}_{pred}^{truth} = \Pr(Class_i) \times \text{IoU}_{pred}^{truth} \tag{3.8}$$

The version of YOLO used in this dissertation is YOLOv3 (Redmon & Farhadi 2018). YOLOv3, shown in Figure 3.6, has a different network for feature extraction than the original YOLO. The network is a hybrid of YOLOv2, Darknet-19 and residual blocks (Redmon & Farhadi 2018), consisting of successive 3×3 and 1×1 convolutional layers and shortcut connections, resulting in a total of 53 convolutional layers (Redmon & Farhadi 2018).

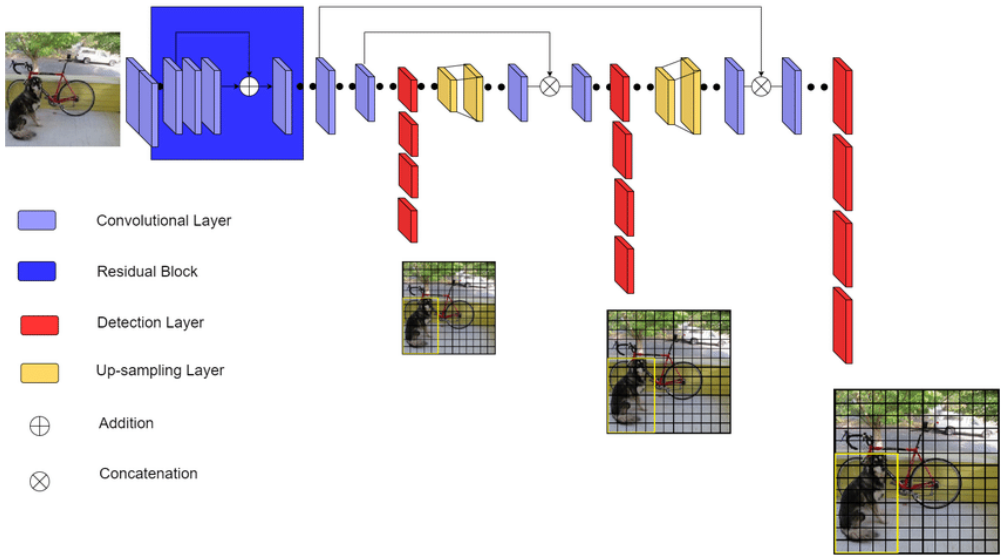


Figure 3.6: Illustration of the YOLOv3 architecture that has 53 convolutional layers, consisting of convolutional layers, residual blocks and detection layers. Source: Valdez (2020)

Claimed advantages of YOLO are its speed, its ability to process the entire image "globally" during training, and a greater ability to generalize (Redmon et al. 2016). However, YOLO does not have as high an accuracy rate as other object detection models, especially when localizing small objects (Redmon et al. 2016). This occurs because strong spatial constraints imposed on bounding box predictions (each grid cell can only predict two boxes , which must share the same class) limits the number of nearby objects the model can predict. Other object detection models, such as Fast R-CNNs, are said to have higher accuracy rates, but lower speed values compared to YOLO (Redmon & Farhadi 2018). Despite these limitations, YOLO has been and continues to be widely deployed in applied object detection research (Jo et al. 2017, Valdez 2020)

3.3.1 Model Evaluation

Mean Average Precision (mAP)

Mean average precision (mAP) is a metric in information retrieval tasks (Salton & McGill 1983) that has been adapted for use in object detection problems since the PASCAL VOC 2007 challenge (Everingham et al. 2010). Average precision was a concept developed to deal with the opposing objectives of recall, the percentage of objects that are correctly predicted, and precision, the percentage of predictions made that are correct. An ideal system would be one that exhibits both high recall and high precision (Salton & McGill 1983). However, techniques that improve recall tend to lower precision and vice-versa. Therefore, researchers may decide what goal is more important for their specific system, i.e., either high recall or high precision (Singhal et al. 2001). This is where average precision comes in because it is known to be a reliable and highly informative performance measure (Salton & McGill 1983). Average precision is the average of precision values for eleven evenly spaced recall values between zero and one, as illustrated on the precision-recall curve in Figure 3.7. Then mAP is the mean of the average precision (AP) for all classes.

For object detection tasks, the true positives, false positives and false negatives are defined using the intersection over union (IoU) ratio. Traditionally, an IoU of greater than 0.5 is needed for a prediction to be considered as a true positive. A false positive occurs when the IoU is less than 0.5 or there are duplicate detections. A false negative occurs when no detection is made though the target object is present. The IoU threshold of 0.5 was deliberately set low to account for inaccuracies in bounding boxes in the ground truth data (Everingham et al. 2010). For example, defining the bounding box for highly nonconvex objects can be slightly subjective (Everingham et al. 2010). In this dissertation, an example would be whether or not to include the swimming tentacles of jellyfish, illustrated in Figure 3.8.

AP summarises the shape of the precision-recall curve

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r) \tag{3.9}$$

where $p_{interp}(r)$ is the precision value that is interpolated from the maximum precision measured for which the corresponding recall, $\tilde{r}$, exceeds r:

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{3.10}$$

where $p(\tilde{r})$ is the measured precision at recall, $\tilde{r}$

The interpolation of the precision value is done to remove the effect of the "wiggles" that are caused by small variations in the ranking of detections (Everingham et al. 2010). The mAP for object detection is then the average of the APs calculated for all the classes. Most recent calculations of AP (Equation 3.11) are computed as area under the curve by numerical integration. This method improves precision as well as the ability to measure differences between models with low AP (Everingham & Winn 2011).

$$AP = \sum (r_{n+1} - r_n) p_{interp}(r_{n+1}) \tag{3.11}$$

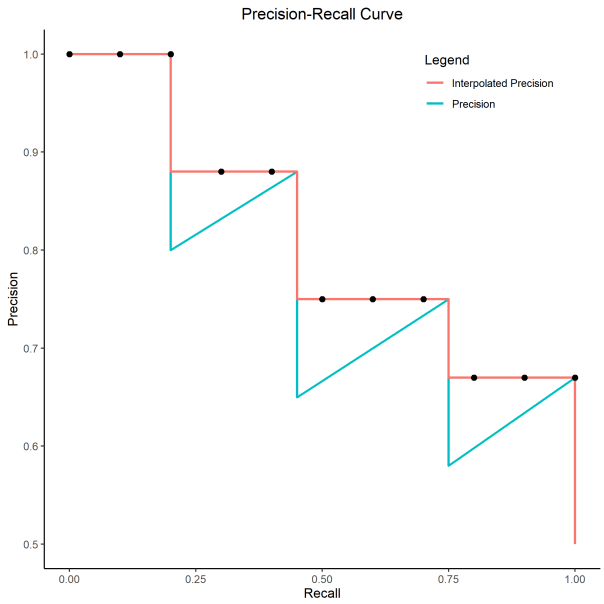


Figure 3.7: The Precision-Recall Curve. The true precision curve (in blue) has "wiggles" caused by small variations in the ranking of detections. The interpolated precision curve (in red) eliminates this problem.

Figure 3.8: Subjective ground truth bounding boxes for an image of a jellyfish, with the red box only including the body of the jellyfish, with the large black box also includes the tentacles of jellyfish

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r}) \tag{3.12}$$

3.4 Deep Learning in Marine or Ecological projects

Deep learning techniques have been successfully applied to many fields, including medicine (Litjens et al. 2017) and agriculture (Kamilaris & Prenafeta-Boldú 2018) and are increasingly being used to solve ecological problems.

Computer vision has been used to address ecological questions such as species recognition (Zhou et al. 2015, Norouzzadeh et al. 2018, Cheng et al. 2017, Gómez et al. 2017), individual animal identification (Polzounov et al. 2016, Beery et al. 2020, Norouzzadeh et al. 2021), animal behavioural classifications (Browning et al. 2017, Norouzzadeh et al. 2018), and biodiversity estimations (Rammer & Seidl 2019). The use of deep learning in ecology is still in its infancy, with many applications still annotated manually (Palei et al. 2016, Alonso et al. 2015), however, as more data is being collected and only a small fraction of it being utilised, it is becoming a priority to analyse these images automatically (Moniruzzaman et al. 2017).

Most deep learning applications in ecology have been in image classification (Rammer & Seidl 2019), with it having been applied to both terrestrial (Tabak et al. 2019) and marine species (Lu et al. 2017). There are also examples of ecological applications of object detection models on both terrestrial (Schneider et al. 2018) and marine species (Zhou et al. 2015, Guirado et al. 2019).

Chapter 4

Methodology

This chapter describes the dataset used in this dissertation and then explores the different methodologies implemented for the classification and detection tasks. Section 4.1 describes the dataset used, followed by Section 4.2, that looks at how the data was extracted for use in both models. Section 4.3 explores the pre-processing steps, the model architectures and software packages used for the image classification problem. Similarly, Section 4.4 focuses on the pre-processing steps, the model architectures and software packages of the object detection problem.

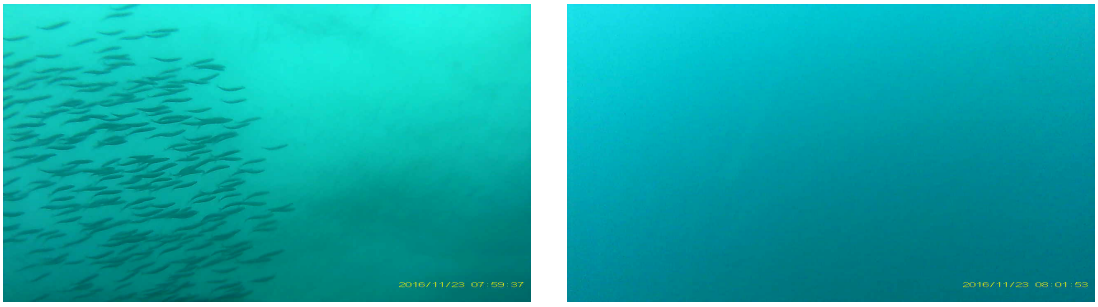
4.1 Data Description

The dataset consists of video footage from animal-borne cameras, mounted on little penguins, see Figure 1.1. The videos focused on the foraging behaviour of the penguins. These amounted to 95 hours of footage, with a typical video being 20 minutes long. The data came from 21 different penguins over a period of 5 and 4 days in October and November 2016 respectively. The data was also accompanied with an annotation file, that was manually complied by a subject expert. The annotation file contains foraging information such things as the type, abundance and capture of prey as well as the presence and abundance of conspecifics or heterospecifics (the presences of other penguins or birds).

4.2 Data Extraction

Using FFmpeg (2019), an image extraction software, the videos were extracted into a series of images. For example, a typical video would be 20-minute long and produce around 36000 images when extraction is done at 30 frames per second (fps). Then the supplied annotation file was used to identify the times within the footage that contained prey events. For the purposes of this study, a prey event is defined as when a known prey of the penguin is seen within a frame. This means that events where the penguin is not seen feeding on the said prey is still counted as a prey event, with a single event spanning several frames. Figure 4.1 shows an example of a prey and non-prey event extracted from the videos. The classes of prey investigated were schools of small fish, that present as "bait balls" (dense spheres of tightly-packed fish), and jellyfish. Only videos that were shown by the annotation file to have

high prey counts were used.



(a) Prey Event (b) Non Prey Event

Figure 4.1: Examples of the 1280 × 720 pixel images extracted from penguin borne videos. (a) part of a school of fish visible that is classed as a prey event, while (b) an empty underwater image of the ocean is classed as a non-prey event.

4.3 Image Classification

4.3.1 Pre-processing Methods

For the image classification task, images containing prey events and an equal number of randomly selected non-prey events were used. The result was a dataset of 6734 images. The dataset was split by videos into train, validation, and test sets with the ratios 70%; 16%, 14% respectively. The data was split into different sets according to the videos the images were obtained from, this is done to avoid images from the same prey event being placed in different sets. Prey events span several images and those images can be almost identical when taken from the same second or short interval. The videos tend to be from different times, at least ten minutes apart or taken on different days, thus ensuring a single event is not split over the train, test or validation sets.

	Train	Validation	Test	Total
Prey	2382	477	508	3367
Non-prey	2311	431	625	3367

Table 4.1: The number of images in each class that made up the train, validation, and test sets. The images were split by video such that similar images were placed in the same set.

The images were then resized to 224 × 224 pixels to be compatible with the default format of ResNet. The dataset was then pre-processed using an inbuilt feature of Keras' ResNet50 API (Chollet et al. 2015) (He et al. 2016). This includes converting images from RGB (red, green, blue) to BGR (blue, green, red), then each colour channel is zero-centered with respect to the ImageNet dataset, without scaling (TensorFlow 2020). Zero-centring is a pre-processing technique that subtracts the mean, in this case the mean pixel value from ImageNet, from each of the data point to make it zero-centered. This is done because most machine learning and deep learning algorithms work better with zero-centred, normalized featured (Chollet

2018).

4.3.2 Neural Network Methods

The output layer of the original ResNet model was removed to adjust the model to fit the objective of this dissertation (Figure 4.2). A fully connected hidden layer, and a final output layer with two neurons were added. The weights of the original architecture were frozen, to avoid losing useful information contained in them (Chollet 2018). Only the new layers were trainable, and had their weights changed during further training rounds.

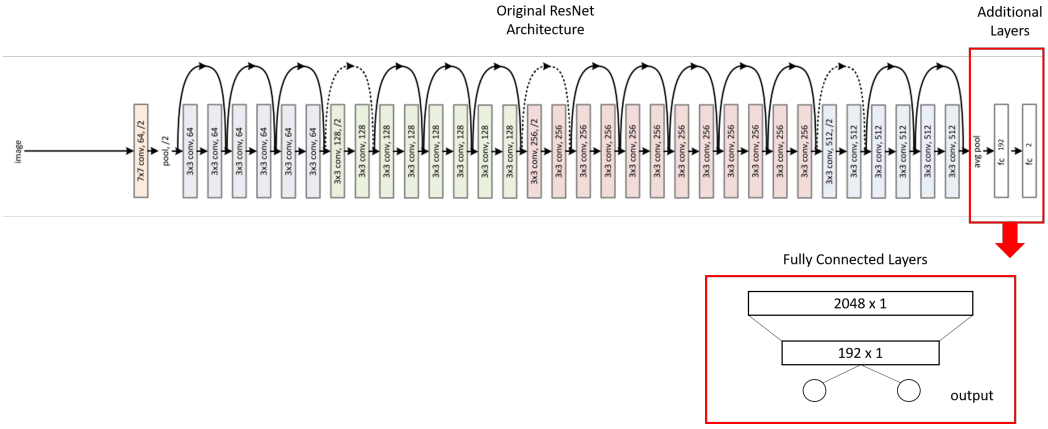


Figure 4.2: ResNet50 architecture (He et al. 2016) adapted with additional fully connected layer, and output layer with two units.

Hyperparameter tuning discussed in Section 4.3.3 was performed on the model to determine optimal hyperparameters. A semi-supervised method was employed next using images from two videos containing prey events that were set aside. These videos were short clips, each containing a prey event over several images, and surrounded by an equal number of non-prey event images. The model makes certain assumptions or generalizations about the different classes of the dataset when making predictions, the semi-supervised methods aim to provide counterexamples to challenge these assumptions (Zhu & Goldberg 2009), as semi-supervised learning methods aim to understand how models behave and how that behaviour changes in the presence of both labelled and unlabelled data (Zhu & Goldberg 2009). These additional videos act as unlabelled unseen data, the classification of which is predicted by the model. Whatever the model misclassified was then added to the training set, and refitted to improve its performance.

The best model was determined based on its accuracy, precision, recall and F1-scores. These values were obtained from its performance on the validation set, before and after the inclusion of the additional images from the semi-supervised task to the training set. Once this is complete, the best model is tested on the test set to see its overall performance.

4.3.3 Hyperparameters

The hyperparameters that were checked include the learning rate, the number of hidden dense layers added to the architecture, the number of neurons in the layers, and dropout rate.

Hyperparameter	Selected values
Learning rate	0.01, 0.001, 0.0001, 0.00001
No. of additional hidden dense layers	1, 2
No. of neurons in additional layers	64, 128, 192, 256
Dropout rate	0.0, 0.2, 0.4, 0.6

Table 4.2: The different hyperparameters tested for the image classification model.

Due to computational constraints, we were unable to search all 128 combinations of hyperparameters in Table 4.2. The combination of hyperparameters tuned was selected by the Keras "Tuner" function. The tuning function "Tuner" allows the user to supply the different hyperparameters, and it performs a search to find the best combination to investigate. The tuning can either be performed as a random search, or by algorithms such as HyperBand. Keras's HyperBand is a modified approach that optimises hyperparameter selection process by utilising early stopping to address resource allocation, that allows it to evaluate more configurations (Li et al. 2018).

4.3.4 Software Packages

As mentioned FFmpeg was used to extract images from video footage. The data was pre-processed, that is, resized and colour normalized, using Python libraries OpenCV (Bradski 2000) and Keras.preprocess_input. The deep learning models were built using Keras APIs (Chollet et al. 2015). All modelling computations were done on Google Colaboratory (Google 2019), a free online cloud computing service with GPU enabled.

4.4 Object Detection

4.4.1 Pre-processing Methods

For the object detection task, only images containing prey events were used. The result was a dataset of 3536 images. Bounding boxes were created for the dataset using a labeling software, LabelImg, that outputs text files with the bounding box coordinates in the necessary format. The jellyfish were labelled individually, while fish were labelled collectively as a school. The dataset was again split by videos into train, validation, and test sets with the ratios 63%; 11%, 26% respectively. Table 4.3 shows that the dataset contained more jellyfish data points than schools of fish.

Prey events are relatively rare in our dataset, but images of schools of fish and jellyfish are abundant online. We hypothesised that adding in externally-collected (open source) images of prey would improve the performance of our classifier. We collected 864 images of both schools of fish and jellyfish, from the open source image library, Open Images Dataset, an

	Train	Validation	Test	Total
Jellyfish	1286	359	741	2386
Schools of Fish	983	90	317	1390

Table 4.3: The number of data points for each prey type that made up the train, validation, and test sets. Images may contain more than one object, such that the sum is greater than the total number of images. The images were again split by video such that similar images were placed in the same set.

online repository of freely accessible and labelled images collected from Flickr (Kuznetsova et al. 2018). The additional data did not have a label for schools of fish, so the label was assigned to fish in order to align with our data. The dataset also contained the objects in various states outside of their habitat or in various forms.

4.4.2 Neural Network Methods

The YOLO model described in Section 3.3 was used for the object detection problem. Hyperparameter tuning discussed in Section 4.4.3 was performed on the model to determine optimal hyperparameters. The best model was assessed based on its mean average precision (mAP) on the validation set.

4.4.3 Hyperparameters

The configuration of some YOLO hyperparameters can be adjusted to improve performance. These include the learning rate, number of layers, and the size of images. The size of the images becomes important when dealing with small objects, for examples balls of fish in the distance.

Due to the long runtime of the model, not all combinations of the hyperparameters were evaluated. One hyperparameter would be changed, while the rest held constant. Initially the values were held constant at their default values, as highlighted in Table 5.7, however, the best performing hyperparameter was used for subsequent hyperparameters. The value with the best result would be assigned to that hyperparameter while another hyperparameter was evaluated. By limiting the hyperparameter combinations, it is possible to have not achieved the optimal solution.

Hyperparameter	Selected values
Learning rate	0.01, 0.001*, 0.0001
Number of YOLO layers	3*, 5
Image Size	416×416*, 608×608, 800×800

Table 4.4: The different hyperparameters that were tested. With the YOLO default values marked with an *

4.4.4 Software Packages

As mentioned FFmpeg was used to extract images from video footage, and LabelImg used to create the bounding boxes. The YOLO model were built using darknet. All modelling

computations were performed on Google Colaboratory, a free online cloud computing service with GPU enabled.

4.5 Test on Full-Length Video

After the performance of the detection and isolation models on test set, it was decided to investigate their ability to predict prey events in an unseen full-length video. This was done to simulate later uses for such models. The same video was used on both models, and pre-processed in accordance with the requirements for the respective models. The video contained three prey events periods of varying lengths, that totalled to 51 images out of 18390 images. These images were labelled by the author, unlike the training, validation and test sets that were labelled using the annotation files provided. Though, all images were presented to the isolation model, the focus will be on the images predicted to contain prey by the detection model.

Chapter 5

Results & Discussion

This chapter presents and discusses the results and observations from the detection and isolation tasks. Section 5.1 looks at the results of the detection task, specifically, the outcomes of the hyperparameter tuning and the performance of the model on unseen supplementary clips and the validation and test sets. Section 5.2 focuses on the isolation task, discussing the results of the hyperparameter tuning and the overall performance of the model. Section 5.3 looks at each model's ability to predict prey events from the images of a full-length video.

The final image classification model was able to achieve 91% on the validation set, and 90% on the test set. The final object detection model was able to achieve mAPs of 63% and 56% on its validation and test sets, respectively. However, similar results were not yielded on the full-length video, with the models yielding a lot of false positives.

5.1 Image Classification

The final model was chosen as the hyperparameters that yielded the highest accuracy. The model had 590,850 trainable parameters and 23,587,712 non-trainable, since it was decided to freeze the layers of the pretrained ResNet-based architecture. This meant that the weights for the frozen layers would not change, i.e., kept their original values from the ImageNet dataset they were trained on.

Table 5.1 summarises the results of the top five performing configurations of the hyperparameter tuning. The accuracies on the validation set range between 88-90%, with the best model achieving an accuracy of 90.75%. This architecture had two additional hidden layers with 256 units each, dropout rates of 0.6 and 0.4 for the first and second layers respectively, and a learning rate of 0.01.

The performance measures of the best performing architectures with different numbers of additional layers are compared in Table 5.2. The model with one additional layer had the third highest accuracy overall. This model also had better precision than the best model with two additional layers, illustrating the challenge in balancing precision and recall.

From the additional videos, the final model was shown to have high specificity and low sen-

Table 5.1: Summary of the top 5 results from hyperparameter tuning of the ResNet-basedImage Classification model on the validation set, in descending order of accuracy.

	accuracy	hidden layer 1	dropout rate	hidden layer 2	dropout rate	learning rate
1	0.9075	256	0.6	256	0.4	0.01
2	0.8921	192	0.2	128	0.2	0.0001
3	0.8899	256	0.4	-	-	0.01
4	0.8888	64	0.4	192	0	0.01
5	0.8855	128	0.2	-	-	0.01

Table 5.2: Comparison of the performance measures of the best performing ResNet-based models with different additional layers.

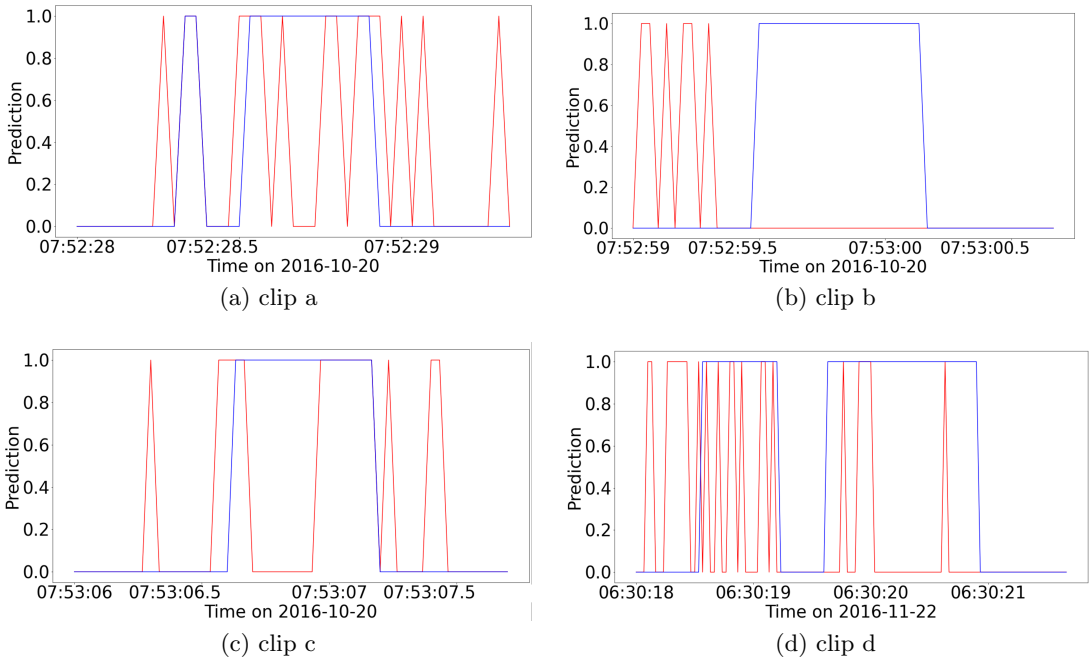
ID	No. of additional layers	accuracy	precision	recall	F1-score
1	2	0.9075	0.8774	0.9455	0.9102
3	1	0.8899	0.9064	0.8931	0.8997

sitivity (Table 5.3). There is usually a trade-off between sensitivity and specificity, so it is expected that when one is high the other is low. Higher specificity suggested the model was better at detecting non-prey events but this may lead to having a lot of false negatives. The performance on the second video was comparable to that of the first video, suggesting the additional data points provided by the first video, did not improve the model's weights significantly. The accuracy achieved on both videos by the model is much lower (51 - 65%) than what was achieved on the validation set. This performance suggesting the model cannot be deployed to automate the prediction prey events. The predictions made by the final model on the additional videos were plotted against their true labels in Figure 5.1. In Figure 5.1, the ground truth graphs indict a temporal relationship between consecutive image frames. Such relationships can be incorporated into deep learning models such as Long-term Recurrent Convolutional Networks (LRCNs) (Conway et al. 2021), however this is beyond the scope of this dissertation.

Table 5.3: The performance measures of the final ResNet-based model on the two additional videos

set	accuracy	sensitivity	specificity	precision	f1-score
Video 1	0.6458	0.3529	0.8065	0.5000	0.4138
Video 2	0.5135	0.2373	0.8269	0.6087	0.3415

Table 5.4 compares the performance metrics of the validation set before and after the inclusion of the additional data (Table 5.3) to the training dataset and refit. It was noted that while the accuracy, specificity and precision of the model on the validation set had improved, the sensitivity had decreased. The improvements in the validation set results also suggest that the model is better at predicting non-prey events.



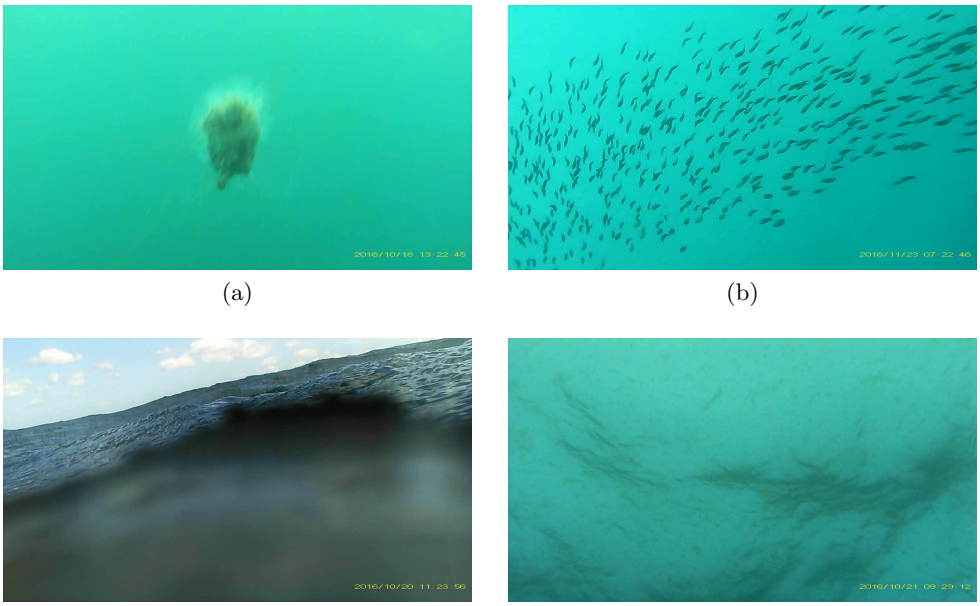
(a) clip a (b) clip b

(c) clip c (d) clip d

Figure 5.1: Plot of prey events predictions (in red),i.e, both jellyfish and schools of fish, by the model compared to the ground truth labels (in blue). Clips a, b and c are different segments from the same unseen video and as such are predicted together, while predictions for clip d were made after the model was retrained to include misclassified images from the previous clips.

Table 5.4: Summary of performance metrics of final trained model on validation set before and after adding semi-supervised learning process.

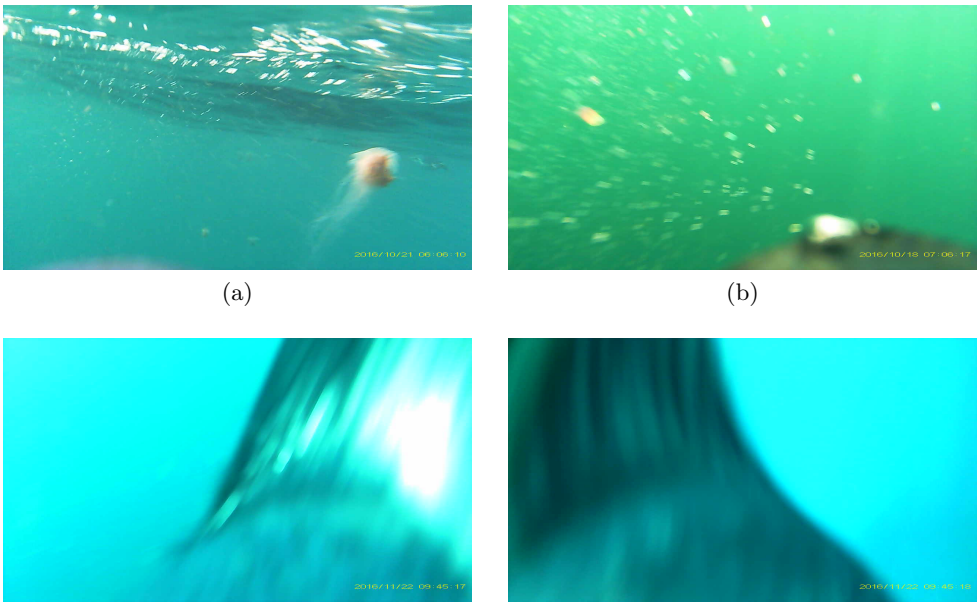
	accuracy	sensitivity	specificity	precision	f1-score
Before	0.88987	0.960168	0.812065	0.84972	0.9016
After	0.91079	0.914046	0.907193	0.91597	0.915



(a) (b)

(c) (d)

Figure 5.2: Images that the detection model was able to identify correctly. Frames (a) and (b) contain prey events, a jellyfish and a school of fish respectively. Frames (c) and (d) were correctly classed non-prey events.



(a) (b)

(c) (d)

Figure 5.3: Images that the detection model was unable to identify correctly. Frame (a) contains a prey event, i.e., a jellyfish, that was classified as a non-prey event. Frame (b) contains air bubbles and debris that was classed as a prey event. Frames (c) and (d) were incorrectly classed prey events when they contain the backs of the penguin carrying the camera.

The performance results obtained on the test set were similar to those obtained on the validation set, as shown in Table 5.5, with high accuracy of 90%, precision of 91% while sensitivity was compromised with a value of 87%. These results are much higher than what was obtained with the additional videos. The data that made up the training, validation and test sets were gathered in a similar way and may have images in a similar setting, which may explain the similar range of results amongst them, while the additional videos may have contained images in environments not previously seen by the model.

Table 5.5: Performance metrics of the test set on the final trained model after the inclusion semi-supervised learning process.

set	accuracy	sensitivity	specificity	precision	f1-score
Test	0.90203	0.86811	0.9296	0.90928	0.8882

Figure 5.2 and 5.3 show examples of correctly and incorrectly classified images, respectively. The model was better able to classify prey events when there was less light variability in the background, and the prey was central to the image. Non-prey events where the sky was visible were also correctly classified (Figure 5.2). The model struggled with images where schools of fish were blurry due to the high turbidity in the water. This might that the schools of fish could easily be mistaken for floating debris or vice versa. The model also misclassified images where the back of the camera-wearing penguin was highly visible (Figure 5.3)

5.2 Object Detection

The results from the hyperparameter tuning on the validation set are documented in Table 5.6. Table 5.7 shows the impact of each hyperparameter on the test set. However, the selection of final hyperparameter values (Table 4.4) was based on the best mAP score on the validation set, because machine learning methods require that all changes made to the model be determined using the training and validation sets, while the test set remains unseen until the end. Due to runtime constraints limited combinations were explored. The learning rate was the first hyperparameter changed, with the best learning rate then being used as the default for subsequent hyperparameter changes. The image size, and number of YOLO layers were also explored, then the impact of open source data was later investigated.

The highest mAP value achieved on the validation set was 63.18%, which was obtained with a learning rate of 0.01, an image size of 416×416 pixels, and 3-layer YOLO configuration (Table 5.6). This hyperparameter conbination resulted in the test mAP being 56.44% (Table 5.7). While the mAP of the validation and test sets may be comparable, the AP of the individual classes differed vastly. For example, the AP of the Jellyfish ranged around 93.4% - 96.2% in the validation set, while in the test set the values were much lower around 48.8% - 56.7%. The high variability was due to the small size of the validation set, especially for the school of fish class, that varied from 30.1% in the validation set to 60.7% in the test set.

Most object detection models achieved mAPs ranging between 44% - 61% on the COCO dataset (Redmon & Farhadi 2018), a large scale image dataset which has 80 classes and 328 000 images, commonly used to benchmark new object detection, segmentation, and caption-

ing algorithms (Lin et al. 2014). Though, this dataset did not have as many classes as the COCO dataset, the model achieved comparable mAP value to that achived by YOLOv3 of 58%.

Increasing the image size was suggested to improve the performance of the model at detecting smaller objects (Bochkovskiy 2018). The default image size was 416×416 pixels, this was increased to 608 and 800. The 800 pixel images performed much worse that the default 416, this is likely due to the original images having dimensions of 1200×700, the model had to compensate the lack of the pixels along the width, and this distorted the images. The 608 pixel images performed slightly worse than the default image size on the validation set, with the AP of the schools of fish dropping from 30% to 6.6%. However, on the test set, the larger images performed better than the default size.

The configuration of the YOLO model that has been used so far, contained a 3-layer YOLO configuration. The library contained another configuration that had a 5-layer YOLO configuration, and this was explored to study its impact on the dataset. It should be noted that the 5-layer YOLO configuration was not run for the full duration needed to reach its completion because the estimated runtime of this configuration was 100 hours, instead it was run for 9hours, the typical duration of the 3-layer YOLO configuration. Results for the 5-layer model were comparable to the 3-layer model. The results of the validation set suggested that the more YOLO layers did not improve the model's preformance, while the test set results suggested that it performed better. It is possible that if the model was allowed to run longer it would have achieved better results than the 3-layer YOLO configuration, however, one would need to balance the improved accuracy with the long runtime.

Despite having several hours of footage, the ratio of prey events to non-prey events was very low, and as such limited the amount of data presented to the model. Thus, the option of additional open source data was added to the training set to investigate its effect. Again, the validation set results suggested that the change did not improve the performance of the model (Table 5.6), while the test set's performance improved from 56.44% to 59.26% mAP with the inclusion of the data (Table 5.7). The additional data contained the objects in various states outside of their habitat or in various forms, such as, animated cartoons, toys, captured by fishermen or displayed as food. These images were not removed and thus may have negatively impacted the way the model understood the objects.

Figure 5.4 and 5.5 show examples of correctly and incorrectly isolated images. The model was able to identify jellyfish in many cases, as shown in Figures 5.4 and 5.5. However, it missed a few jellyfish when the luminance was high. The model also tended to classify other penguins as fish, and sometimes would miss schools of fish completely, as shown in Figure 5.5.

5.3 Performance on full-length Video

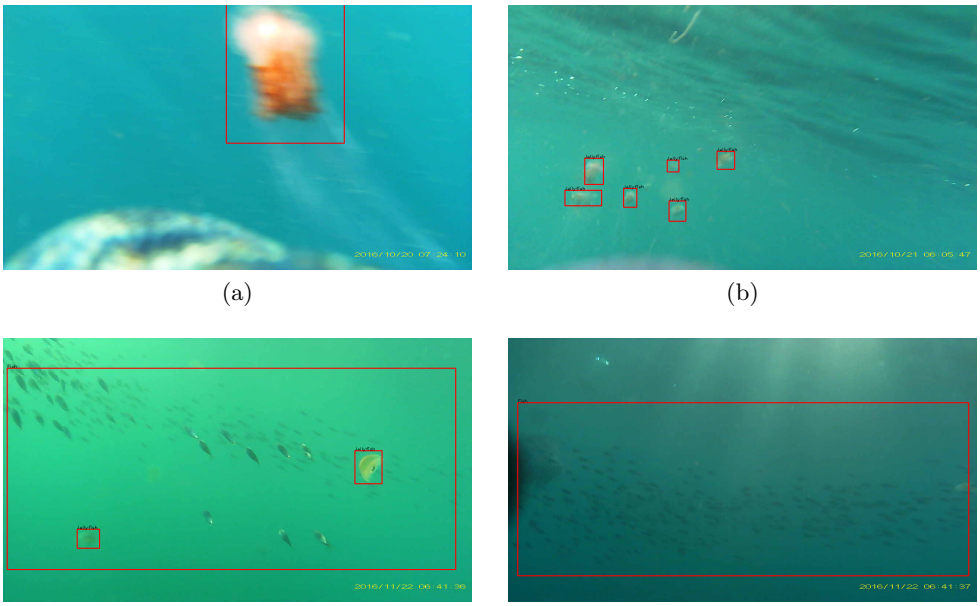
Figure 5.6, shows the ResNet-based model performance on the full-length video. Out of the 51 prey event images, only 4 were correctly identified. Despite this, the model achieved an

Table 5.6: Summary of results from hyperparameter tuning of the YOLOv3 model on the validation set, where * denotes the default value of that hyperparameter, and the best performing value is highlighted in bold

Hyperparameter		Validation Set Results					
Name	Value	Jellyfish AP	Fish AP	mAP	Precision	Recall	F1-score
Learning Rate	0.001 *	95.44	26.81	61.13	0.91	0.77	0.83
	0.01	**96.24**	**30.12**	**63.18**	**0.91**	**0.78**	**0.84**
	0.0001	93.40	14.40	53.90	0.92	0.75	0.82
Image Size	**416 × 416 ***	**96.24**	**30.12**	**63.18**	**0.91**	**0.78**	**0.84**
	608 × 608	96.12	6.64	51.38	0.89	0.77	0.83
	800 × 800	34.78	0.00	17.39	0.65	0.25	0.37
YOLO Layers	**3 ***	**96.24**	**30.12**	**63.18**	**0.91**	**0.78**	**0.84**
	5	95.16	24.32	59.74	0.94	0.73	0.82
Additional Data	**No Data***	**96.24**	**30.12**	**63.18**	**0.91**	**0.78**	**0.84**
	Open Source Data	96.37	20.55	58.46	0.92	0.78	0.84

Table 5.7: Summary of results from hyperparameter tuning of the YOLOv3 model on the test set, where * denotes the default value of that hyperparameter.

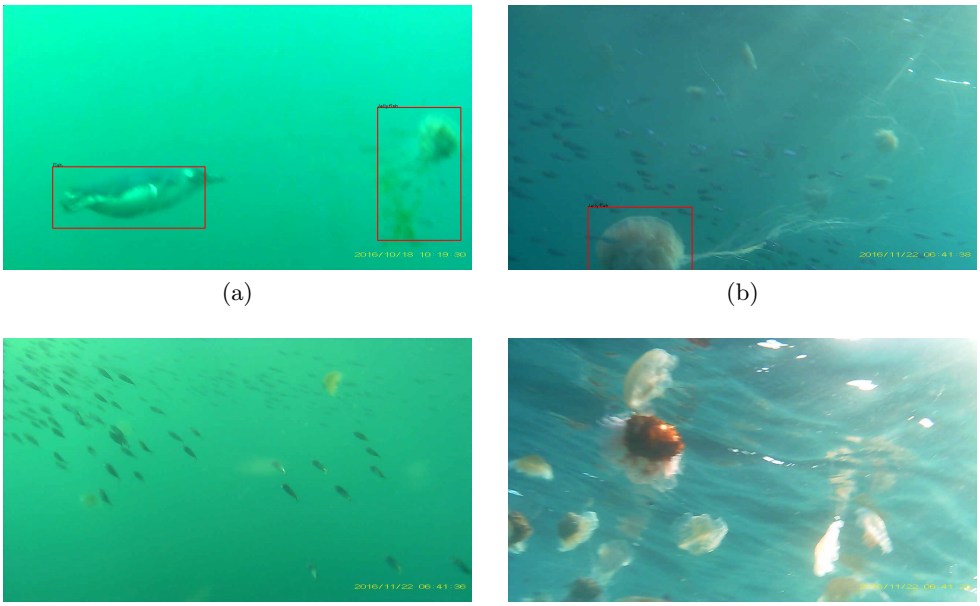
Hyperparameter		Test					
Name	Value	jellyfish AP	fish AP	mAP	Precision	Recall	F1-score
Learning Rate	0.001 *	48.75	57.35	53.05	0.82	0.35	0.49
	0.01	52.21	60.67	56.44	0.81	0.41	0.55
	0.0001	53.93	56.11	55.02	0.80	0.41	0.54
Image Size	416 × 416 *	52.21	60.67	56.44	0.81	0.41	0.55
	608 × 608	56.74	59.23	57.99	0.84	0.46	0.60
	800 × 800	18.53	0.00	9.27	0.69	0.12	0.21
YOLO Layers	3 *	52.21	60.67	56.44	0.81	0.41	0.55
	5	56.59	61.15	58.87	0.87	0.4	0.55
Additional Data	No Data*	52.21	60.67	56.44	0.81	0.41	0.55
	Open Source Data	59.18	59.35	59.26	0.84	0.42	0.56



(a) (b)

(c) (d)

Figure 5.4: Images that the isolation model was able to identify correctly. Frames (a) and (b) contain jellyfish, while frames (c) and (d) contain schools of fish respectively.



(a) (b)

(c) (d)

Figure 5.5: Images that the isolation model was unable to identify correctly. Frame (a) contains a jellyfish and penguin which the model labeled as a fish. Similarly, in frame (b) the model identified the jellyfish, however, missed the school of fish that was also present. While no detection were made in frames (c) and (d) though the images contains a school of fish and a group of jellyfish respectively.

accuracy of 95%, however, its precision and recall were 0.005 and 0.078 respectively. As mentioned previously, the model had high specificity which suggests it would be better able to predict non-prey events, this in combination with the hugely unbalanced data, resulting in a high accuracy but low recall. Therefore the model has too many false positives for it to be used in place of manual annotation.

These results were in contrast with what was achieved by the test set. Since the training, validation and test sets contained equal numbers of prey and non prey images, a lot of non-prey images were discarded to create balanced sets, i.e., undersampled non-prey class. The sampling process which was done manually, may have been biased in favour to certain types of non-prey events, for example, images with less ambiguity. By not including all non-prey images, the model may have struggled when presented a full-length video with all types of non-prey events.

The images that the ResNet-based model predicted to contain prey events were presented to the YOLO model. From the 805 images, the YOLO model drew bounding boxes in 641 images, and classed these as prey events (Figure 5.7) and then eliminated the other 164 images as non-prey events. The accuracy of the models when used together becomes 96%, however, one of the prey event images is discarded by the YOLO model. For the 4 true prey events, 1 was not detected and did not have a bounding box. While the other 3 were correctly isolated into bounding boxes, 1 image had a school of fish misclassified as a jellyfish. The remaining 2 images were correctly labelled and isolated.

When we presented all images from the full-length video to the YOLO-based model, it was able to detect bounding boxes in 29 of the 51 prey event images. Overall, the model achieved a classification accuracy of 73%. Upon interrogation, it was noted that other penguins appearing in frames were being misclassified as jellyfish or fish. Penguins were originally ignored in the training of the model. Therefore, the model was retrained to include a penguin label to explore its impact, however, this did not improve the models performance on the full-length video, as shown in Figure 5.8. The inclusion of the penguin hindered the model's recall and precision abilities, with recall decreasing from 0.569 to 0.176 and precision decreasing from 0.006 to 0.002. The penguins and prey were being misclassified as each other. For example the shape of a swimming penguin with its flippers by its side was similar to that of the jellyfish. In both cases, it was noted that penguins and circular light reflection in the water tended to be misclassified as jellyfish, while debris, air bubbles in the water and crashing waves tended to be misclassified as fish.

The YOLO model also served as a detection check on the predictions from the ResNet-based model, by eliminating some false positives when it did not detect an object to draw a bounding box around.

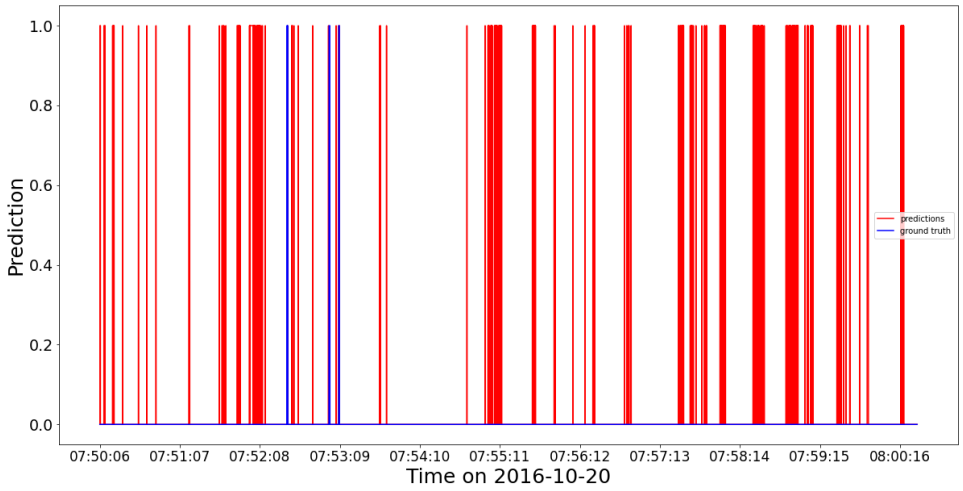


Figure 5.6: ResNet-basedmodel's predictions (in red) of prey events on a full-length video compared to the ground truth labels (in blue). The 51 frames containing prey are distributed over 4 distinct prey events (blue vertical bars).

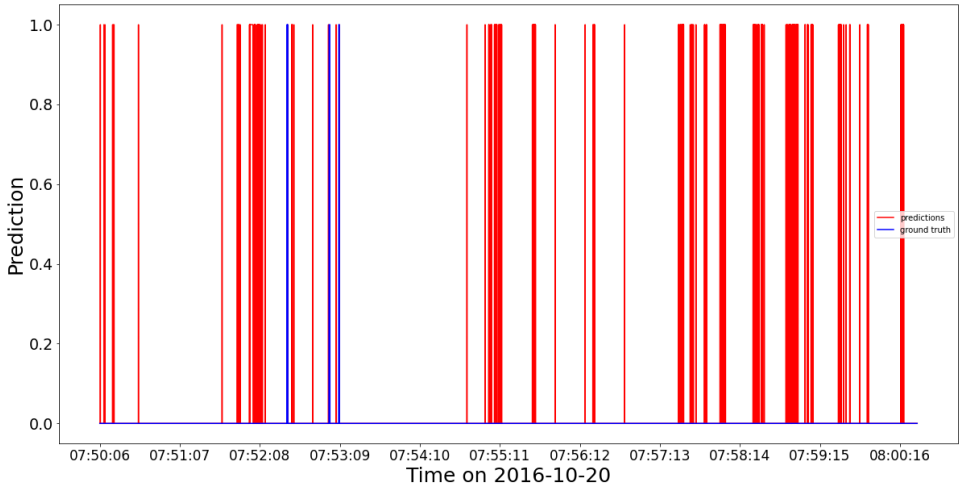
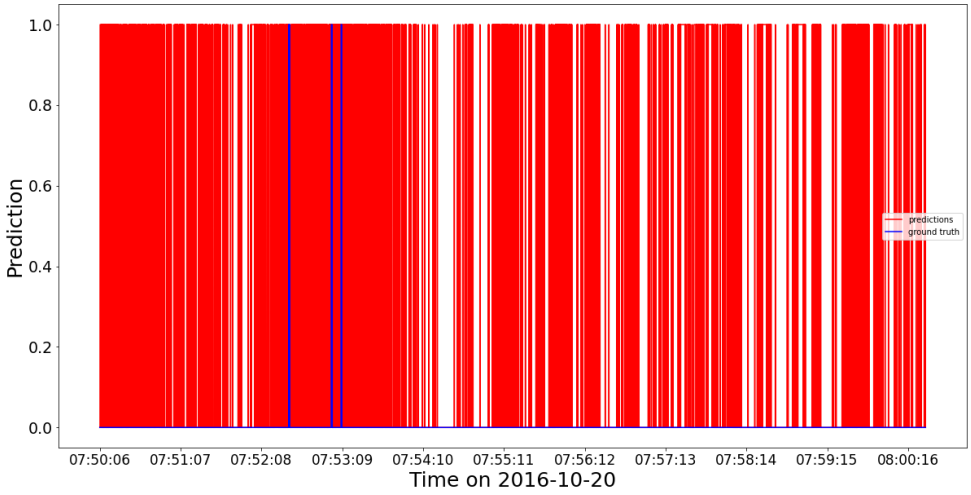
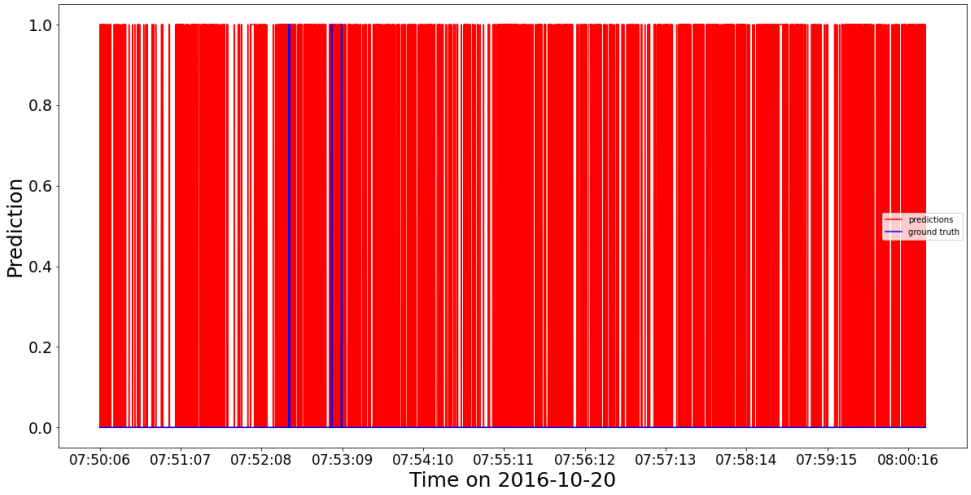


Figure 5.7: YOLO model's predictions (in red) of prey event bounding boxes on predicted prey events from ResNet-based model, compared to the ground truth labels (in blue).The 51 frames containing prey are distributed over 4 distinct prey events (blue vertical bars).



(a) YOLO prediction on unseen video before the inclusion of the penguin label. Results shown to contain a large number of false positives.



(b) YOLO prediction on unseen video after the inclusion of the penguin label. Results shown to contain even more false positives, as prey animals are confused for penguins.

Figure 5.8: YOLO model's predictions (in red) of prey event bounding boxes on a full-length video compared to the ground truth labels (in blue). The 51 frames containing prey are distributed over 4 distinct prey events (blue vertical bars).

Chapter 6

Conclusions

The objective of this dissertation was to develop models that would detect and isolate prey events from footage captured by animal-borne cameras, specifically, from little penguins. These models would ideally function to assist researchers by automating annotating image files, which is currently a laborious task. This is a result of the high volumes of data being produced due to advances in data collection technology such as animal-borne cameras. The speed at which data is produced greatly outpaces the speed of methods currently available to process and extract information from the data.

Independent models were built to tackle each task, i.e., detection and isolation. An image classifier based on on the ResNet architecture was developed for the detection task. This model achieved 91% and 90% accuracies on the validation and test sets respectively. The model had high specificity signalling it was better at detecting non-prey events than prey events, with 92% of the true non-prey events being correctly identified as such. The model used for the isolation task was a YOLO object detector. On the validation and test sets, the model was able to achieve mean average precision (mAP) values of 63 and 56, respectively.

The models were then used in series on an unseen full length video to simulate real world usability. However, the detection model did not perform as well on images from unseen video segments. The sensitivity and precision of the model were very low, with the model identifying 805 images as prey events with only 4 being correct. These 805 images were used as the input of the isolation model. The model drew bounding boxes in 641 of these images, while missing 1 of the 4 true prey events and mislabelling a school of fish as a jellyfish in 1 of the other images. This result highlighted the poor performance of the isolation model.

The research objective was to help researchers automatically annotate their videos. The outcomes of this dissertation suggest that these models cannot serve this purpose on their own. However, the models can be used to highlight promising areas of concern in the videos which would speed up processing time.

One avenue for future research, would be a model that performs both functions, i.e., detection and isolation. Utilising negative sampling, to present an object detection model with a balanced dataset of prey and non-prey events, could achieve this objective. Output images with

no bounding boxes and labels would act as non-prey predictions, while, those with bounding boxes would be classified as prey event predictions.

Some of the limitations discovered during this dissertation can be used to facilitate other future works related to this topic. These include the following:

1. **Data augmentation** - The data used in both models was not augmented. Augmentation increases the size of the dataset, and may assist the models to generalize better, having seen prey in many orientations.

2. **Selective supplementary data** - It might be useful to supplement the dataset with more relevant data than that provided by Google's open images dataset, ideally open-source images from other ecological projects.

3. **Negative samples** - This means including images that do not contain objects of interest in an object detector model, to assist in the learning of the model. The high false positive results of YOLO model on the full length video may be attributed to the fact that the model was not exposed to non-prey events (Bochkovskiy 2018).

4. **"Sequence Voting"** - In some cases it was noted that several images may seem identical yet be placed into different classes by the model. To reduce fluctation in predictions, it might be useful for the model to consider a sequence of images before making a prediction.

Appendices

Appendix A

Research Code

The research code to this dissertation, Detection and Isolation of Prey Capture Events in Animal-Borne Images, can be found in the GitHub repository link below:

https://github.com/Temweka/Isolation_and_Detection_Prey

Bibliography

Aggarwal, C. C. (2018), *Neural Networks and Deep Learning*, Springer International Publishing, Cham.

Alonso, R., Mcclintock, B., Lyren, L., Boydston, E. & Crooks, K. (2015), 'Mark-recapture and mark-resight methods for estimating abundance with remote cameras: A carnivore case study', *PLoS ONE* .

Beery, S., Wu, G., Rathod, V., Votel, R. & Huang, J. (2020), Context r-cnn: Long term temporal context for per-camera object detection, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 13075–13085.

Beijbom, O., Edmunds, P. J., Roelfsema, C., Smith, J., Kline, D. I., Neal, B. P., Dunlap, M. J., Moriarty, V., Fan, T.-Y., Tan, C.-J. et al. (2015), 'Towards automated annotation of benthic survey images: Variability of human experts and operational modes of automation', *PloS one* **10**(7), e0130312.

Bergstra, J. & Bengio, Y. (2012), 'Random search for hyper-parameter optimization.', *Journal of machine learning research* **13**(2).

Bochkovskiy, A. (2018), 'Darknet (neural network framework)-wiki'.
URL: *https://github.com/AlexeyAB/darknet/wiki*

Bradski, G. (2000), 'The OpenCV Library', *Dr. Dobb's Journal of Software Tools* .

Browning, E., Bolton, M., Owen, E., Shoji, A., Guilford, T. & Freeman, R. (2017), 'Predicting animal behaviour using deep learning: Gps data alone accurately predict diving in seabirds', *Methods in Ecology and Evolution* **9**, 681–692.

Brownlee, J. (2019), *Deep Learning for Computer Vision*, Springer.

Burns, G. L. (2006), 'The fascination of fur and feathers: managing human-animal interactions in wildlife tourism settings', *Australian Zoologist* **33**, 446–457.

Camprasse, E. C. M., Sutton, G. J., Berlincourt, M. & Arnould, J. P. Y. (2017), 'Changing with the times: little penguins exhibit flexibility in foraging behaviour and low behavioural consistency', *Marine Biology* **164**, 1–10.

Cannell, B. & Cullen, J. M. (1998), 'The foraging behaviour of little penguins eudyptula minor at different light levels', *IBIS* **140**, 467–471.

Cheng, X., Zhang, Y., Chen, Y., Wu, Y.-Z. & Yue, Y. (2017), 'Pest identification via deep residual learning in complex background', *Computers and Electronics in Agriculture* **141**, 351–356.

Chicco, D. & Jurman, G. (2020), 'The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation', *BMC genomics* **21**(1), 1–13.

Chollet, F. (2018), *Deep Learning with R*, Manning Publications Co.

Chollet, F. et al. (2015), 'Keras'. GitHub.
URL: *https://github.com/fchollet/keras*

Christin, S., Hervet, É. & Lecomte, N. (2019), 'Applications for deep learning in ecology', *Methods in Ecology and Evolution* **10**(10), 1632–1644.

Clevert, D., Unterthiner, T. & Hochreiter, S. (2016), Fast and accurate deep network learning by exponential linear units (ELUs), *in* '4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings'.

Conway, A. M., Durbach, I. N., McInnes, A. & Harris, R. N. (2021), 'Frame-by-frame annotation of video recordings using deep neural networks', *Ecosphere* **12**(3), e03384.

Csáji, B. C. et al. (2001), 'Approximation with artificial neural networks', *Faculty of Sciences, Eötvös Loránd University, Hungary* **24**(48), 7.

Dann, P. (1992), 'Distribution, population trends and factors influencing the population size of the little penguins 'eudyptula minor' on phillip island', *Emu* **91**, 263–272.

Doyle, T. K., Hays, G. C., Harrod, C. & Houghton, J. D. (2014), Ecological and societal benefits of jellyfish, *in* 'Jellyfish blooms', Springer, pp. 105–127.

Duchi, J., Hazan, E. & Singer, Y. (2011), 'Adaptive subgradient methods for online learning and stochastic optimization.', *Journal of machine learning research* **12**(7).

Erhan, D., Szegedy, C., Toshev, A. & Anguelov, D. (2014), Scalable object detection using deep neural networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2147–2154.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J. & Zisserman, A. (2010), 'The pascal visual object classes (voc) challenge', *International journal of computer vision* **88**(2), 303–338.

Everingham, M. & Winn, J. (2011), 'The pascal visual object classes challenge 2012 (voc2012) development kit', *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep* **8**, 5.

FFmpeg Developers (2019), 'ffmpeg tool'. Version 4.2 [Software].
URL: *http://ffmpeg.org/*

Flemming, S. A., Lalas, C. & Heezik, Y. v. (2013), 'Little penguin (eudyptula minor) diet at three breeding colonies in new zealand', *New Zealand Journal of Ecology* **37**, 199–205.

Girshick, R. (2015), Fast r-cnn, *in* '2015 IEEE International Conference on Computer Vision (ICCV)', pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, *in* '2014 IEEE Conference on Computer Vision and Pattern Recognition', pp. 580–587.

Glorot, X., Bordes, A. & Bengio, Y. (2011), Deep sparse rectifier neural networks, *in* 'Proceedings of the fourteenth international conference on artificial intelligence and statistics', JMLR Workshop and Conference Proceedings, pp. 315–323.

Gómez, A., Salazar, A. & Vargas-Bonilla, J. F. (2017), 'Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks', *Ecological Informatics* **41**, 24—32.

Goodfellow, I., Bengio, Y. & Courville, A. (2016), *Deep learning*, MIT press.

Google (2019), 'Google colaboratory'.
URL: *http://colab.research.google.com*

Guirado, E., Tabik, S., L. Rivas, M., Alcaraz-Segura, D. & Herrera, F. (2019), 'Whale counting in satellite and aerial images with deep learning', *Scientific Reports* **9**.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hubel, D. H. & Wiesel, T. N. (1962), 'Receptive fields, binocular interaction and functional architecture in the cat's visual cortex', *The Journal of physiology* **160**(1), 106–154.

James, G., Witten, D., Hastie, T. & Tibshirani, R. (2013), *An Introduction to Statistical Learning*, Springer, New York.

Jo, K., Im, J., Kim, J. & Kim, D.-S. (2017), A real-time multi-class multi-object tracker using yolov2, *in* '2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)', IEEE, pp. 507–511.

Kamilaris, A. & Prenafeta-Boldú, F. X. (2018), 'Deep learning in agriculture: A survey', *Computers and electronics in agriculture* **147**, 70–90.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**, 1097–1105.

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A. et al. (2018), 'The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale', *arXiv preprint arXiv:1811.00982* .

Lebrato, M., Pitt, K. A., Sweetman, A. K., Jones, D. O., Cartes, J. E., Oschlies, A., Condon, R. H., Molinero, J. C., Adler, L., Gaillard, C. et al. (2012), 'Jelly-falls historic and recent observations: a review to drive future research directions', *Hydrobiologia* **690**(1), 227–245.

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

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

LeCun, Y., Touresky, D., Hinton, G. & Sejnowski, T. (1988), A theoretical framework for back-propagation, *in* 'Proceedings of the 1988 connectionist models summer school', Vol. 1, pp. 21–28.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. (2018), 'Hyperband: A novel bandit-based approach to hyperparameter optimization', *Journal of Machine Learning Research* **18**, 1–52.

Lin, T.-Y., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L. (2014), Microsoft coco: Common objects in context, *in* 'ECCV'.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., Van Der Laak, J. A., Van Ginneken, B. & Sánchez, C. I. (2017), 'A survey on deep learning in medical image analysis', *Medical image analysis* **42**, 60–88.

Lu, H., Li, Y., Uemura, T., Ge, Z., Xu, X., He, L., Serikawa, S. & Kim, H. (2017), 'Fdcnet: filtering deep convolutional network for marine organism classification', *Multimedia Tools and Applications* **77**, 21847–21860.

Maas, A. L., Hannun, A. Y., Ng, A. Y. et al. (2013), Rectifier nonlinearities improve neural network acoustic models, *in* 'Proc. icml', Vol. 30, Citeseer, p. 3.

McCann, S. & Reesman, J. (2013), 'Object detection using convolutional neural networks'.

McCarthy, J. (2007), 'Basic questions: What is artificial intelligence?'. [Online] Accessed: 2020-10-12.
URL: *http://www-formal.stanford.edu/jmc/whatisai/node1.html*

McCarthy, J., Minsky, M. L., Rochester, N. & Shannon, C. E. (2006), 'A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955', *AI magazine* **27**(4), 12–14.

McInnes, A. M., McGeorge, C., Ginsberg, S., Pichegru, L. & Pistorius, P. A. (2017), 'Group foraging increases foraging efficiency in a piscivorous diver, the african penguin', *Royal Society open science* **4**(9), 170918.

Medsker, L. R. & Jain, L. (2001), 'Recurrent neural networks', *Design and Applications* **5**, 64–67.

Mhaskar, H., Liao, Q. & Poggio, T. (2016), 'Learning functions: when is deep better than shallow', *arXiv preprint arXiv:1603.00988* .

Moniruzzaman, M., Islam, S., Bennamoun, M. & Lavery, P. (2017), Deep learning on underwater marine object detection: A survey, *in* 'International conference on advanced concepts for intelligent vision systems (ACIVS)', pp. 150–160.

Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jojic, N. & Clune, J. (2021), 'A deep active learning system for species identification and counting in camera trap images', *Methods in Ecology and Evolution* **12**(1), 150–161.

Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C. & Clune, J. (2018), 'Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning', *Proceedings of the National Academy of Sciences* **115**(25), E5716–E5725.

Palei, H. S., Pradhan, T., Sahu, H. & Nayak, A. (2016), 'Estimating mammalian abundance using camera traps in the tropical forest of similipal tiger reserve, odisha, india', *Proceedings of Zoological Society* **69**, 181–188.

Pan, S. J. & Yang, Q. (2009), 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering* **22**(10), 1345–1359.

Papageorgiou, C. P., Oren, M. & Poggio, T. (1998), A general framework for object detection, *in* 'Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)', IEEE, pp. 555–562.

Parker, C. (2013), 'On measuring the performance of binary classifiers', *Knowledge and Information Systems* **35**(1), 131–152.

Polzounov, A., Terpugova, I., Skiparis, D. & Mihai, A. (2016), 'Right whale recognition using convolutional neural networks', *ArXiv* **abs/1604.05605**.

Preston, T., Ropert-Coudert, Y., Kato, A., Chiaradia, A., Kirkwood, R., Dann, P. & Reina, R. (2008), 'Foraging behaviour of little penguins eudyptula minor in an artificially modified environment', *Endangered Species Research* **4**, 1–9.

Rammer, W. & Seidl, R. (2019), 'Harnessing deep learning in ecology: An example predicting bark beetle outbreaks', *Frontiers in Plant Science* **10**, 1327.

Rawat, W. & Wang, Z. (2017), 'Deep convolutional neural networks for image classification: A comprehensive review', *Neural computation* **29**(9), 2352–2449.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779–788.

Redmon, J. & Farhadi, A. (2018), 'Yolov3: An incremental improvement', *arXiv preprint arXiv:1804.02767* .

Ren, S., He, K., Girshick, R. & Sun, J. (2017), 'Faster r-cnn: Towards real-time object detection with region proposal networks', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **39**(6), 1137–1149.

Ropert-Coudert, Y., Kato, A., Wilson, R. P. & Cannell, B. (2006), 'Foraging strategies and prey encounter rate of free-ranging little penguins', *Marine Biology* pp. 139–148.

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

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al. (2015), 'Imagenet large scale visual recognition challenge', *International journal of computer vision* **115**(3), 211–252.

Sagar, R. (2019), 'What does freezing a layer mean and how does it help in fine tuning neural networks', Analytics India Magazine. Online; accessed 10 January 2021.
URL: *https://analyticsindiamag.com/what-does-freezing-a-layer-mean-and-how-does-it-help-in-fine-tuning-neural-networks/*

Salton, G. & McGill, M. J. (1983), *Introduction to modern information retrieval*, mcgraw-hill.

Schindler, A., Lidy, T. & Rauber, A. (2016), Comparing shallow versus deep neural network architectures for automatic music genre classification., *in* 'Proceedings of the 9th Forum Media Technology (FMT)', pp. 17–21.

Schneider, S., Taylor, G. W. & Kremer, S. (2018), Deep learning object detection methods for ecological camera trap data, *in* '2018 15th Conference on computer and robot vision (CRV)', IEEE, pp. 321–328.

Simonyan, K. & Zisserman, A. (2014), 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

Singhal, A. et al. (2001), 'Modern information retrieval: A brief overview', *IEEE Data Engineering Bulletin* **24**(4), 35–43.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* **15**, 1929–1958.

Sutton, G. J., Hoskins, A. J. & Arnould, J. P. Y. (2015), 'Benefits of group foraging depend on prey type in a small marine predator, the little penguin', *PLoS ONE* **10**, 1–10.

Sze, V., Chen, Y.-H., Yang, T.-J. & Emer, J. S. (2017), Efficient processing of deep neural networks: A tutorial and survey, *in* 'Proceedings of the IEEE', Vol. 105, Ieee, pp. 2295–2329.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1–9.

Szegedy, C., Toshev, A. & Erhan, D. (2013), Deep neural networks for object detection, *in* 'Advances in Neural Information Processing Systems', Vol. 26, Curran Associates, Inc., pp. 2553–2561.

Szeliski, R. (2011), *Computer Vision*, Springer.

Tabak, M., Norouzzadeh, M. S., Sweeney, S., Vercauteren, K., Snow, N., Halseth, J., Salvo, P., Lewis, J., White, M., Teton, B., Boughton, R., Wight, B., Newkirk, E., Odell, E., Brook, R., Moeller, A., Mandeville, E., Clune, J., Miller, R. & Schlichting, P. (2019), 'Machine learning to classify animal species in camera trap images: Applications in ecology', *Methods in Ecology and Evolution* **10**, 585–590.

TensorFlow (2020), 'tf.keras.applications.resnet.preprocess_input'. [Online] Accessed: 14/10/2020).
URL: *https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet/preprocess_input*

Thiebot, J.-B., Arnould, J. P., Gómez-Laich, A., Ito, K., Kato, A., Mattern, T., Mitamura, H., Noda, T., Poupart, T., Quintana, F. et al. (2017), 'Jellyfish and other gelata as food for four penguin species–insights from predator-borne videos', *Frontiers in Ecology and the Environment* **15**(8), 437–441.

Tieleman, T., Hinton, G. et al. (2012), 'Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude', *COURSERA: Neural networks for machine learning* **4**(2), 26–31.

Valdez, P. (2020), 'Apple defect detection using deep learning based object detection for better post harvest handling', *arXiv preprint arXiv:2005.06089* .

Verma, G. K. & Gupta, P. (2018), Wild animal detection using deep convolutional neural network, *in* 'Proceedings of 2nd international conference on computer vision & image processing', Springer, pp. 327–338.

Vieira, S., Pinaya, W. H. & Mechelli, A. (2017), 'Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications', *Neuroscience & Biobehavioral Reviews* **74**, 58–75.

Wang, L., Chu, F. & Xie, W. (2007), 'Accurate cancer classification using expressions of very few genes', *IEEE/ACM Transactions on computational biology and bioinformatics* **4**(1), 40–53.

Weiss, K., Khoshgoftaar, T. M. & Wang, D. (2016), 'A survey of transfer learning', *Journal of Big data* **3**(1), 1–40.

Wu, J. (2017), 'Introduction to convolutional neural networks', *National Key Lab for Novel Software Technology. Nanjing University. China* **5**(23), 495.

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

Zachariah, N., Kothari, S., Ramamurthy, S., Osunkoya, A. O. & Wang, M. D. (2014), Evaluation of performance metrics for histopathological image classifier optimization, *in* '2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society', IEEE, pp. 1933–1936.

Zhou, H., Llewellyn, L., Wei, L., Creighton, D. & Nahavandi, S. (2015), Marine object detection using background modelling and blob analysis, *in* '2015 IEEE International Conference on Systems, Man, and Cybernetics', pp. 430–435.

Zhu, X. & Goldberg, A. B. (2009), 'Introduction to semi-supervised learning', *Synthesis lectures on artificial intelligence and machine learning* **3**(1), 1–130.