Photo-Identification of Marine Cetaceans Using Convolutional Neural Networks

Cameron Trotter

School of Electrical and Electronic Engineering

Newcastle University

In Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

SUBMISSION DATE

DEDICATION

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated in the text. This dissertation contains fewer than 50,000 words inclusive of footnotes but excluding appendices and bibliography.

Cameron Trotter
SUBMISSION DATE

Acknowledgements

ACKNOWLEDGEMENTS

Abstract

ABSTRACT

Table of contents

List of figures **xiii**

List of tables **xv**

1 Introduction **1**
1.1 Research Problem . . . 3
1.2 Contributions . . . 3
1.3 Thesis Structure . . . 3
1.4 Related Publications . . . 3

2 Background **5**
2.1 A Brief Introduction to Photo-Identification . . . 5
2.2 A Brief Introduction to Deep Learning . . . 8
2.2.1 Supervised and Unsupervised Learning . . . 10
2.2.2 The Stochastic Gradient Descent Algorithm . . . 11
2.2.3 Backpropagation . . . 12
2.3 Deep Learning for Computer Vision . . . 13
2.3.1 Convolutional Neural Networks . . . 13
2.3.2 Object Detection . . . 17
2.3.3 Semantic Segmentation . . . 19
2.3.4 Part Segmentation . . . 22
2.3.5 Instance Segmentation . . . 22
2.3.6 Fine-Grained Visual Categorisation . . . 26
2.4 Computer Vision for Conservation Technology . . . 27
2.4.1 Utilising Computer Vision in Cetacean Conservation . . . 28
2.5 Summary . . . 32

3 Cetacean Detection Using Deep Learning **33**
3.1 Requirements of a Cetacean Detector . . . 33

3.1.1 Environmental Requirements 34
3.1.2 Technical Requirements 36
3.2 Deciding on Architecture and Framework 37
3.2.1 An Investigation into Bounding Boxes 38
3.2.2 Instance Segmentation Architectures 41
3.3 Initial Testing of Mask-RCNN 42
3.3.1 The Zanzibar Dataset 43
3.3.2 Transfer Learning 45
3.3.3 Utilising Transfer Learning to Train the Mask-RCNN 46
3.3.4 Data Augmentation 47
3.3.5 Mask-RCNN Hyperparameter Tuning 49

Appendix **53**

Bibliography **55**

List of figures

1.1 Map of the survey area, Northumberland, UK, from St. Mary's Lighthouse in the south to 25nm above Coquet Island in the north. 2

2.1 Examples of the main body parts utilised in cetacean photo-id. Left: callosities present on the head of a northern right whale (*Eubalaena glacialis*) [118]. Right: fluke of a humpback whale (*Megaptera novaeangliae*) [26]. Bottom: dorsal fin of a common bottlenose dolphin (*Tursiops truncatus*) [154]. . . . 7

2.2 An example neural network architecture diagram, showing VGG16 [143]. Data is inputted to the network at the leftmost layer before being passed through the layers sequentially. The rightmost layer provides the overall output of the network, such as a classification or probability. [62] 9

2.3 A visual representation of convolution. A kernel, represented by the grey squares, operates over a padded input matrix, blue, with a stride of 2 to produce an output feature map, green. Note that the kernel's weights are denoted by the number in the lower right of each box and the input pixel value is denoted by the number in the middle of each box. [40] 14

2.4 A visualisation of the ReLU Activation Function 16

2.5 An example of RoIs generated on an image by an RPN, showing 10 random proposals out of the 200 generated. Note that two of the RoIs have been successfully classified as `fin`. 18

2.6 An example of an image and it's corresponding ground truth `fin` masks. . . 20

2.7 Generated anchors. Left: negative anchors. Middle: neutral anchors. Right: positive anchors. 21

2.8 An example of refined anchors. Positive anchors before refinement are dotted, after refinement are solid. 21

2.9 An example showing the difference between semantic and instance segmentation. In semantic segmentation, all pixels which belong to the `person` class have been classified as one `person` object. In instance segmentation, five `person` objects have been detected, and all `person` pixels have been assigned to one of the objects. Image from [141]. 23
2.10 The Faster R-CNN architecture [129]. The blue box represents operations in stage one. The green box represents operations in stage two. 25
2.11 A visual representation of the changes made to Faster R-CNN to create Mask R-CNN. [64] 26

3.1 Some cetaceans, such as bottlenose dolphins, travel in pods. The developed detection system must be capable of splitting this pod into individual animals to be passed to the identifier. 34
3.2 Two images of the same individual taken from different angles of approach, directions of travel, and distances from the vessel. Note how this changes the make-up of the dorsal fin, however keeps the identifying notch visible. 35
3.3 An example manual crop utilised in bounding box suitability testing. . . . 38
3.4 An example manual crop showing the result of SIFT feature extraction when thresholded based on RGB colour values. 40
3.5 An example manual crop showing the result of GrabCut background removal. 41
3.6 An example image showing the labelling processes using VIA. 45

List of tables

Chapter 1

Introduction

Modelling cetacean (whale, dolphin, and porpoise) population dynamics and behaviour is paramount to effective population management and conservation. Robust data is required for the design and implementation of conservation strategies and to assess the risks presented by anthropogenic activity such as offshore wind turbines and commercial fishing. Moreover, cetaceans make prime candidates for modelling ecosystem change under the ecosystem sentinel concept as they reflect the current state of the ecosystem and respond to change across different spatial and temporal scales [105]. As the global climate changes and urbanisation of coastal areas intensifies, it is imperative to develop methodologies for quick and effective assessment of the biological and ecological impact of rising sea temperatures, pollution, and habitat degradation. This can be achieved through modelling the population, behaviour, and health of large marine species such as dolphins.

Methodologies of cetacean research includes photo identification (photo-id). Photo-id involves collecting photographic data and identifying individuals based on unique permanent markings, and has been used for more than 40 years for modelling cetacean population dynamics and ecology [33, 176]. Current identification techniques for cetaceans rely heavily on experts manually identifying individuals. This can often be costly due to the number of person-hours required for identification, as well as the large potential for error due to issues such as observer fatigue. Further, individual identification of dolphins within a species is time consuming due to the nature of the task. Intra-species dolphins have very similar markings and body types making identifying an individual within a pod very difficult. Prominent features must be identified, such as small nicks to the fins or scars left from injuries to identify an individual. If these features are only prominent on one side of the individual, the task of identification becomes even more difficult.

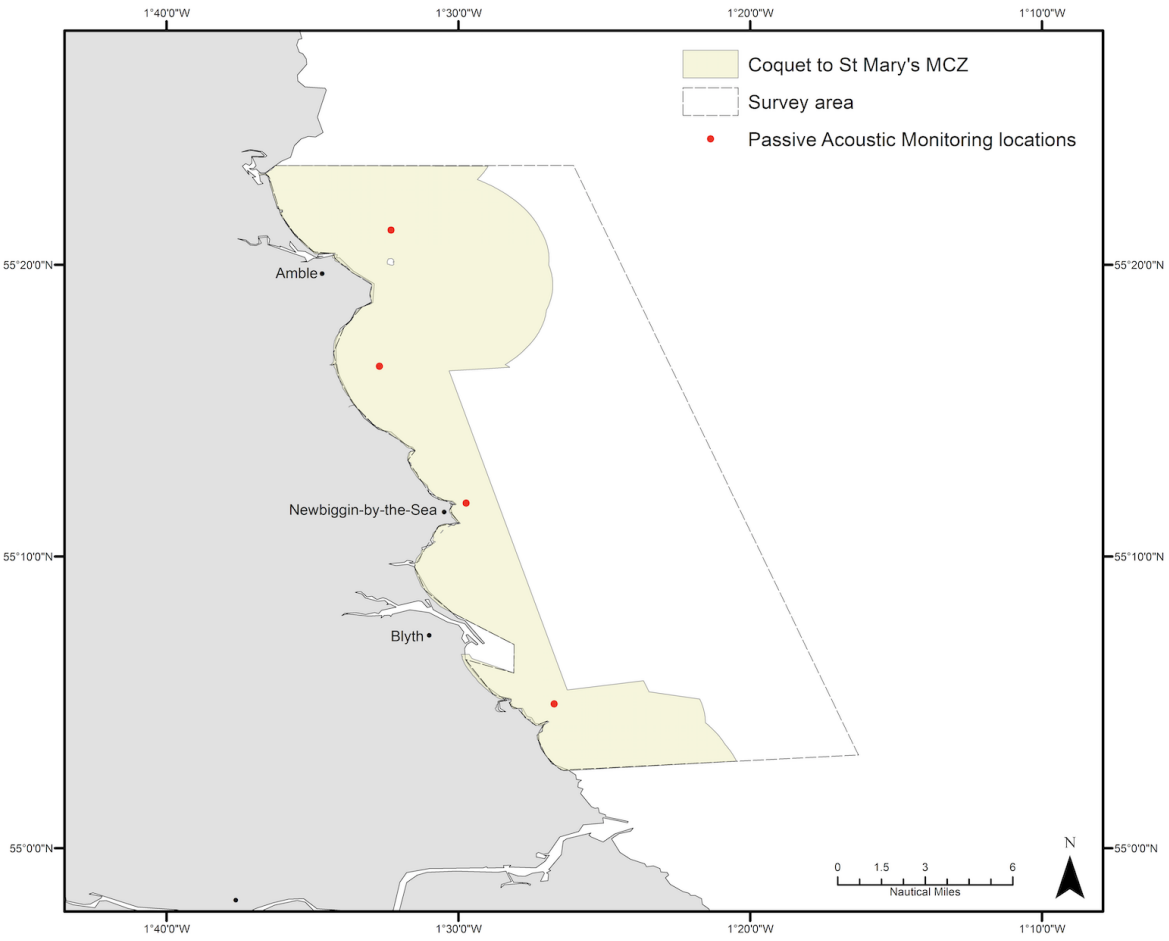

Figure 1.1 Map of the survey area, Northumberland, UK, from St. Mary's Lighthouse in the south to 25nm above Coquet Island in the north.

With progressively more data being collected during fieldwork through increased use of technology, there is an urgent need for an automatic system for quick identification with reduced error rates. Previous efforts to photo-id individuals from underwater video stills from previous expeditions undertaken by Newcastle University's School of Natural & Environmental Science's Marine MEGAfauna Lab took around three months from raw video file to be completely catalogued. This project addresses these limitations by applying the methodologies, techniques, and computational power of deep learning to the field of marine biology. Deep learning models, specifically Convolutional Neural Networks (CNNs), will be trained on high-end computer clusters using the Microsoft Azure Cloud[1] prior to field studies using existing data. Once trained, the models can be ran on field deployable computers to perform image analysis in real time from multiple data sources (underwater and above water

[1]Microsoft Azure Cloud: azure.microsoft.com

images, and aerial drone footage). Methodologies incorporating these models are designed to quickly identify individuals, assess health, analyse behaviour and incorporate remote sensing techniques.

CNNs have for many years now been regarded as the main approach for solving image and computer vision related problems. More recently, the development of deep-layered CNNs and the availability of high-powered GPUs have provided the perfect platform for solving fine-grained computer vision tasks. This project has developed a system to speed up marine cetacean photo-id using a pipeline of CNNs. Starting with a large high resolution image, this pipeline allows for the detection and identification of cetaceans in the image. This system can greatly aid marine biologists, speeding up the identification process allowing for more time to be spent on developing response strategies and health assessments.

Data collection for this project focussed on a population of white-beaked dolphins (*Lagenorhynchus albirostris*) off the coast of North-East England (see Figure 1.1). Recent research has identified sites where the species is regularly sighted [49, 59] and underwater image analysis has shown seasonal and multi-year residency. A health assessment based on underwater image analysis identified high incidence of skin disease and trauma suggesting conservation of this population should be high priority [159]. The species would also serve as a prime sentinel for monitoring North Sea climatic changes as it shows preference for cold water with North-East UK coastal waters representing the southern limit of its range.

1.1 Research Problem

1.2 Contributions

1.3 Thesis Structure

1.4 Related Publications

Chapter 2

Background

In recent years, deep learning has become a widely used technique within the world of computing. Deep learning models, those comprised of a large number of processing layers in order to learn multiple abstracted representations of the data provided, have consistently been shown to outperform other machine learning techniques, especially in the field of computer vision [81]. As the field of deep learning increases in scope every day, this Chapter will focus primarily on the world of deep learning in a computer vision context. A brief introduction to deep learning will be provided, before looking at how these methodologies have been applied to computer vision, specifically Convolutional Neural Networks (CNNs) and their use in object detection. Literature focusing on computer vision in a marine cetacean space will be explored, as well as the current space of fine-grained computer vision. As these novel technologies are applied to photo-identification, an introduction to this technique is provided for context.

2.1 A Brief Introduction to Photo-Identification

One of the main goals of conservation research is to monitor resident animal populations in a given geographic area. This is most commonly performed using mark-recapture surveys in which researchers identify the number of unique individuals in an area at a given time, before returning to the same area at a later point in time and again identifying the number of individuals present. These values allow for an estimate of the total population size to be obtained, with the accuracy of this value increasing proportionally to the number of surveys undertaken. These mark-recapture surveys are classified as either invasive, where animals are physically trapped, tagged, and released, or non-invasive where monitoring is performed passively.

Photo-identification, often abbreviated to photo-id, is one of the main non-invasive mark-recapture methods utilised by cetacean researchers, usually undertaken over large geographic areas at sea through the use of a small boat although monitoring from coastlines or aircraft may also be utilised [47, 117, 175]. More recently, the use of citizen science has also began to be incorporated within photo-id surveys where evidence exists of smaller species population densities over large areas which may make full-scale monitoring infeasible [29, 51].

Initially utilised for the tracking of individual distinctive animals within a species [24, 139], the methodology was quickly adapted to large-scale monitoring of whole pods. Photo-id is primarily utilised during cetacean health monitoring and abundance estimation studies, with proven use cases on a variety of cetacean species such as Indian Ocean humpback dolphins (*Sousa plumbea*) [142], Risso's dolphins (*Grampus griseus*) [104], Northern bottlenose whales (*Hyperoodon ampullatus*) [43], and killer whales (*Orcinus orca*) [13]. Outside of cetaceans, photo-id has further found use studying other marine life such as whale sharks (*Rhincodon typus*) [69], sea turtles (both *Chelonia mydas* and *Eretmochelys imbricata*) [128], and Florida manatees (*Trichechus manatus latirostris*) [80].

As can be seen from the examples of species where photo-id is utilised, this methodology for mark-recapture relies on the species having some form of individually identifiable markings, similar to human fingerprints. Typically, this identifying information is located on a part of the body which is likely to breach the water at some point during an encounter, although examples of underwater photo-id do exist in literature though are not yet commonplace [161]. Depending on the species of animal, different parts of the body are the primary identifying location; for dolphins this is usually the dorsal fin whilst for whales this is primarily the fluke, or callosities if present [4, 6, 34, 142, 162]. See Figure 2.1 for examples.

During photo-identification surveys, researchers will often focus on long lasting markers such as body-part shape, nicks, notches, and pigmentation which have been shown to be stable throughout the life of the animal [94, 101, 176]. In some cases secondary markers, those which may heal and are thus not stable such as scarring, may also be utilised for identification. These secondary markers may be anthropogenic, for example from collision with a vessel, or natural, for example from encounters with prey. Scarring is of particular use when identifying Risso's dolphins who are well known for the persistent nature of their scarring, which is thought to occur due to a loss of pigmentation when their scars heal [102]. Pigmentation also occurs in other cetacean species such as striped dolphins (*Stenella coeruleoalba*) and has been used for photo-identification where it can be considered a primary marker [134].

Regardless of the species being analysed or the body-parts used during the photo-id process, some assumptions must be universally made. For one, all of the markers must be

Figure 2.1 Examples of the main body parts utilised in cetacean photo-id. Left: callosities present on the head of a northern right whale (*Eubalaena glacialis*) [118]. Right: fluke of a humpback whale (*Megaptera novaeangliae*) [26]. Bottom: dorsal fin of a common bottlenose dolphin (*Tursiops truncatus*) [154].

considered stable, that is, they must not fade over the years. Even if a photo-id study only occurs over a few years, the markers utilised must be stable enough so that if another survey is conducted in the same area in later years, individuals from the first study must still be identifiable, providing useful information to health assessments, population estimates, and residency surveys. This stability reduces false negatives, where one individual is recorded as multiple over time. Second, the markers must be considered individually unique. Those chosen to identify an individual must not overlap with other individuals in the survey area. This reduces the chance of false positives, where multiple individuals are recorded as one. Chosen markers must also allow for a consistent re-sighting probability over time. This is critical for abundance estimates, ensuring that an individual's chosen markers providing it with the same chance of being spotted one year as another. As such, it is extremely important that photo-id methodologies are standardised, both at an international level and between researchers in the same organisations. This process began in 1988 through workshops held by the International Whaling Commission, with further recommendations published in 2015 by Urian *et al.* [60, 158].

Because of the assumptions which must be adhered too, as well as the manual nature of the photo-id process, there are many downsides. Being able to identify individuals relies on high quality photographs. Thanks to the advent of digital photography and the relative inexpensiveness of cameras capable of capturing large megapixel images, this is less of an issue than before, although it still must be considered. Surveys can also only be undertaken

in good weather conditions in terms of sea state and lighting, both of which can affect the chance of an accurate match. These conditions are harder to meet in some areas of the world, reducing the suitability of photo-id for some geographic areas. Conditions, as well as the nature of the animal itself, may make photographing both sides of the individual impossible. Markings are rarely duplicated on both sides of an individual, and thus not having both sides may make matching difficult. For example, an individual may have an extremely distinctive marking on the left side of their dorsal fin however if only the right side of the individual has been captured, when the left side is also eventually photographed it may be labelled as a new individual as no previous examples of the individual's left side exist in the catalogue.

Furthermore, photo-id as a whole is extremely labour intensive. Unlike land based camera trap systems, marine based photo-id surveys require a large human effort. Staff are needed not just for photographic purposes, but also for piloting of vehicles. As the surveyed animals are free roaming, large spatial areas must be covered, and there is no guarantee of encountering them during a given day. Back on land, the captured photographs must then be manually worked through and the individuals in them identified. This can often take longer than the entire data collection period. Thanks to the labour intensiveness of the photo-id process, it can also be extremely costly to undertake. Staff need to be paid, vehicles need to be fuelled, and equipment must be maintained. Because of this, any solutions which may speed up the photo-id process would be welcomed both by researchers and their funding bodies.

2.2 A Brief Introduction to Deep Learning

Deep learning, a subfield of machine learning, aims to create artificial networks to complete tasks, i.e. learn, in a similar way to how neurons in the human brain operate. These computational models are often multiple neurons deep, known as layers, with lower layers representing basic abstractions, building up from this as you go *deeper* into the network, resulting in final layers of neurons which, based on information passed to them from lower levels, can begin to provide estimations of answers to a given problem. In literature, neural networks are often illustrated in forms similar to Figure 2.2, which shows a visual representation of the VGG16 architecture [143].

This ability to learn directly from the data provided is the key difference between deep learning and more classical machine learning techniques, which often require considerable domain expertise to design a feature extractor allowing for raw data values, such as pixels, to be transformed into a feature vector suitable for a model to learn.

Deep learning models in contrast are capable of learning to performing tasks such as classification on raw data values through multiple layers of simple non-linear transformations.

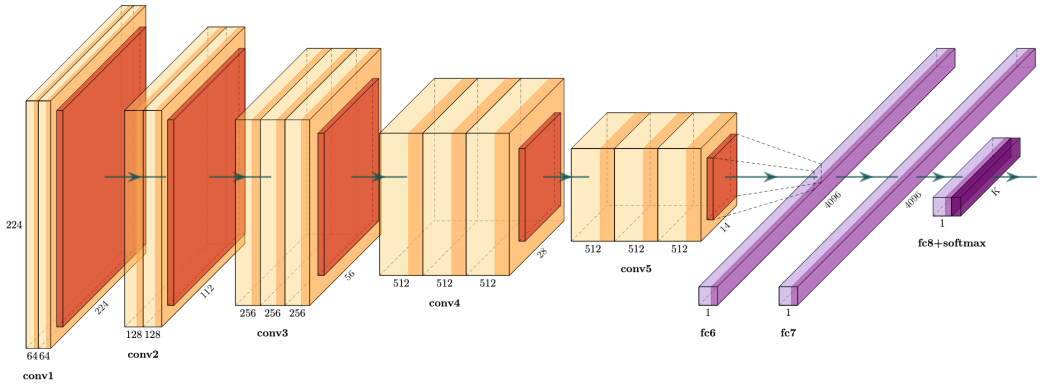

Figure 2.2 An example neural network architecture diagram, showing VGG16 [143]. Data is inputted to the network at the leftmost layer before being passed through the layers sequentially. The rightmost layer provides the overall output of the network, such as a classification or probability. [62]

For example in the case of computer vision, lower layers of neurons may be optimised to learn lines and basic shapes, middle layers may be optimised to learn more complex ideas such as how these lines and shapes fit together, with the final layers providing an output of object label (e.g. `dolphin`). It should be stressed however that the features these layers are looking for are not specified by humans, but rather learned from the data by updating their weights and biases, predominantly through optimisations such as stochastic gradient descent and backpropagation [67]. This is very similar to how the brain learns from the multi-modal data it receives from the body, capturing the intricacies of massive amounts of data using a connection of smaller optimised neurons.

This ambition to create networks similar to how the brain operates stems mainly from work undertaken in 1943 by McCulloch and Pitts [103] in an attempt to understand how neurons in the brain allow for the understanding of complex patters. This model formed the basis of future work into machine learning, and thus deep learning. This work continued at small scale for many years. It has only been recently thanks to advances in availability and power of the computing resources available has deep learning research accelerated. The transition away from model training on CPUs to multiple high-powered GPUs has been one of the largest advancements in deep learning, allowing for a significant speed-up in the train time for these models, resulting in much more prototyping in a smaller time frame. Further to this, the advent of cloud computing has allowed for much more cost-effective model development. Thanks to the Pay As You Go model of computing now commonplace, cloud providers such as Microsoft, Amazon, and Google have eliminated the need for researchers to procure their own hardware required for training, which can in some cases be prohibitively expensive.

More so, advances in deep learning have been helped greatly through the development of standard development frameworks. Google's Tensorflow [1] and Facebook's PyTorch [116] allow for researchers to develop models much faster than previously due to their reduction in the amount of boiler-plate code needed, with these frameworks often doing a lot of the *heavy lifting* in the background. Further advances have been made through the availability of high quality coarse-grained datasets such as MNIST [82], Caltech-256 [57], and ImageNet [37] allowing for common baselines to be adopted by the computer vision community and for the introduction of transfer learning, allowing for the reuse of models trained in one task to be utilised for another [115]. Furthermore, additional regularisation techniques have provided improvements to model accuracy. Notable examples of this in literature which are now commonplace in deep learning models include dropout [145], batch normalisation [72], stochastic gradient descent with warm restarts [96], mixup [182], as well as various forms of data augmentation.

2.2.1 Supervised and Unsupervised Learning

Within machine learning as a whole, all tasks can be placed into two categories, supervised or unsupervised learning.

Supervised learning tasks are those where prior knowledge of what the output value should be, also known as a ground truth, is known. This learning is thus often performed for classification, mapping inputs to a defined set of outputs, or in regression where an input is mapped to a continuous output function. As such, supervised learning's goal can be seen as attempting to understand and generalise the relationship between a set of inputs and a set of outputs.

This generalisation is performed by splitting the available data into training and test sets, with the former being used to train the model and the latter being used to test the model's performance on previously unseen data. Both the training and test set contains ground truth data, but only the training set's influences the generalisation of the model. For example, in the case of a dog or cat classifier a dataset may contain 1000 images, some labelled as `dog` and some as `cat` (the ground truths). This data will then be split randomly into a training and test set; common splits often allocate 80-90% of the data for training with the remaining set used for testing. The classifier will then iterate through the training set, using the ground truth values along with algorithms such as stochastic gradient descent (discussed in more detail in Section 2.2.2) to train each neuron's weights and biases in a way to best generalise the model. After training has been completed the model will then be evaluated against the test set, with each of the unseen images being classified. These predicted classifications will

then be compared to the unseen ground truths in order to provide an evaluation metric of the model's performance.

Unsupervised learning tasks are, in contrast, those where prior ground truths for your data are not available. As such, no classification can be performed using this method, however it is often used for clustering and aiding in understanding of the underlying data structure. These unsupervised algorithms, such as K-Means clustering [63], are not given any guidance on how to define the data into clusters but are rather left to discover interesting structure on their own.

Taking our dog and cat data again as an example, it is clear how this data could be clustered in an unsupervised manner. Asking the clustering algorithm to provide two clusters for the data (e.g. $K = 2$) , a model could be trained to split the data with all dogs in one cluster and all cats in the other, without having to be told which images are dogs and which are cats.

2.2.2 The Stochastic Gradient Descent Algorithm

In order to generalise our deep learning models, we need to be able to optimise the weights and biases within each individual neuron. Most commonly, this is performed using gradient descent to minimise a loss function (a measure of distance between ground truth and prediction) in such a way that weights are updated in the *opposite* direction of the gradient of the loss function. As such, we follow the direction of the slope of the surface created by the loss function downhill iteratively until we reach a minima, an area where the loss is lowest [137]. Before the advent of deep learning and big data, it was common for the whole dataset to be used to compute the gradient at each iteration; however due to the size of modern day datasets this is no longer possible due to the computational cost this would impose on the system. As such, batches of the dataset are often used to give an estimation of the overall loss gradient.

In order to reach this minimum, a process known as stochastic gradient descent (SGD) is most commonly used. At each iteration of SGD, a batch will only contain one randomly selected training example. The loss for this example is calculated, and used to step down the gradient slope, rather than the sum of the loss' gradient over all training examples. As we only take one example per iteration, the path taken down the slope to the minima is far noisier and random than the path obtained from using all examples, hence the *stochastic* nature of the gradient descent. This stochastic nature does result in a longer convergence time to the minima compared to a regular gradient descent, however this is outweighed by the reduction in computational expense. The use of SGD often leads to a good set of model weights quickly compared to other, more elaborate techniques [17].

In recent years there have been efforts to modify SGD in an attempt to improve model optimisation. The most commonly seen optimisations within production code include SGD with warm restarts [96], Momentum [121], RMSProp [153], Adam [76], and AMSGrad [125]. All of these optimisations attempt to stop the problem of getting stuck in local minima rather than the global minimum of the overall loss function. However, recent studies show that the problem of local minima is not as big as first thought and, regardless of initial conditions, vanilla SGD rarely gets stuck in local minima [31, 36].

2.2.3 Backpropagation

As discussed in Section 2.2.2, we have seen how weights and biases in each neuron can be learnt and optimised using SGD. However, it is important that we also discuss how the gradient is computed. This computation can be performed relatively quickly using backpropagation, or the backward propagation of errors algorithm. Before delving *too deep* into deep learning, it is of imperative importance to understand backpropagation; it is after all often cited as one of the cornerstones of deep learning [2].

Backpropagation was originally described in Linnainmaa's masters thesis [90], although it's effect was not fully realised until 1986, when Rumelhart *et al.* discussed the advantages to using backpropagation over other learning approaches [138]. Whilst multiple attempts have been made to improve the original algorithm [10, 83, 87, 88, 113], these are rarely adopted by deep learning researchers as little improvement can be gained from them compared to the overhead of modifying existing deep learning frameworks to incorporate the changes.

The standard backpropagation algorithm to compute the gradient of the loss function w.r.t a model's layers is, in essence, the chain rule (a formula for computing the derivative of multiple functions). Working backwards through the network, the last layer's gradient is first calculated providing a partial calculation of the overall network's gradient. This is then used to efficiently calculate the layer above's gradient, propagating information regarding the loss and how weights should be changed throughout the network. This backwards propagation is a far more computationally efficient way of calculating the overall loss gradient compared to calculating each layer's gradient loss in isolation. Further efficiencies have been made thanks to deep learning frameworks implementing backpropagation in a way that takes advantage of GPUs, leading to extremely efficient computations when performing deep learning tasks such as object detection and other computer vision tasks.

2.3 Deep Learning for Computer Vision

The field of computer vision, allowing computers to gain and interpret knowledge from image and video data, is one area where deep learning has excelled. Generalisable concepts such as CNNs have quickly become commonplace for solving computer vision tasks, in most cases replacing the need for hand-crafted pipelines specialised to the task at hand, thanks to their ability to learn complex patterns in data where there is a strong spatial and temporal dependency between the values. This ability is essential for processing image data which is, at it's most basic level, a matrix of pixel values. These matrices are three dimensional, representing an image's height, width, and depth, where depth is dependant on the colour model used to represent the image. The most common of these models is RGB which has channels representing the red, green, and blue colour present in an image; as such a matrix representing an RGB image will have a depth of three. Other colour models have varying depth values, a greyscale image would have a depth of one for example, whilst a CMYK image would have a depth of four. As can be imagined, these matrices can very quickly reach massive sizes. An RGB image at 1080p resolution for example would require a matrix of size 1920x1080x3, or 6,220,800 values. CNNs allow us to reduce the image down to a more workable form whilst still retaining key features which will allow us to infer predictions.

2.3.1 Convolutional Neural Networks

Modern CNNs are composed of three main layer types; convolutional layers, pooling layers, and fully connected layers. Each of these layers will perform some operation on the input matrix passed to it, and provide a transformed output to the subsequent layer(s). These layers can be stacked in various orientations to build different CNN architectures.

Convolutional Layers

The convolutional layer is the workhorse of the CNN, performing the vast majority of the operations required. Convolutional layers utilise what is known as a kernel in order to efficiently extract features from an input image matrix. A kernel is a matrix, most commonly 3x3 or 5x5 in size, which represents some weighting refined through training. This kernel is slid over the input data computing the dot product between the weights and the section of the input matrix it is currently over, before summing these into a single value. This gives an output matrix of features represented by the weighted sum of the input, known as a feature map. These feature maps generally represent basic shapes at shallow levels, which are the built on as the model gets deeper, representing more complex shapes as you progress.

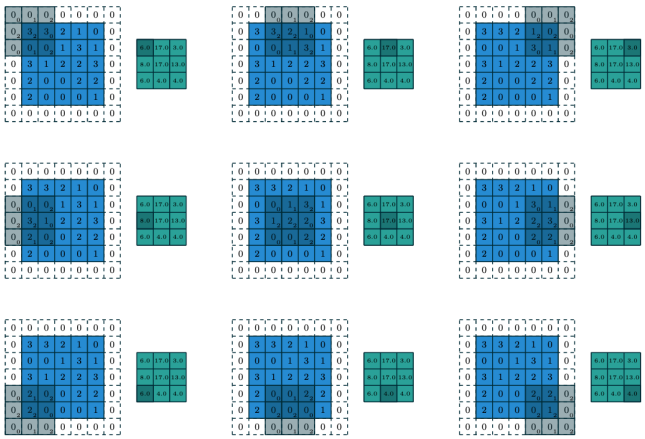

Figure 2.3 A visual representation of convolution. A kernel, represented by the grey squares, operates over a padded input matrix, blue, with a stride of 2 to produce an output feature map, green. Note that the kernel's weights are denoted by the number in the lower right of each box and the input pixel value is denoted by the number in the middle of each box. [40]

The kernel process can be performed multiple times over the same input using different weightings, giving multiple output feature maps. In the case where there are multiple input dimensions (such as an RGB image), kernels are required to operate over all dimensions. As such, the resulting feature maps are summed element-wise, along with some bias term, to produce a single output map.

The size of the kernel as such determines the number of input features which are combined to give the new output feature map, although the size of the resulting map is determined also by two other properties; stride and padding. Stride refers to the distance in pixels the kernel will move when performing the next input mapping. For example, a stride of 1 would result in the kernel sliding along one pixel value each time, resulting in an output feature map of equal size to the input, whereas a stride of 2 would skip every other pixel, reducing the output feature map by half.

A problem can arise during convolution when the kernel reaches the edges of the input matrix. As there is nothing past the edge values, these values must be trimmed as they are never in the centre of the kernel. This can cause the reduction of the matrix, as some values are never utilised, which may be detrimental when we require an output feature map which is the same size as that of the input. As such, padding can be performed. This technique places zeros around the edges of the input matrix, expanding it's size and allowing pixel values formerly at the edges to be utilised by the kernel. A visual representation of how convolution is performed can be seen in Figure 2.3.

Pooling Layers

Pooling layers help reduce the computational complexity of the convolutions performed by the CNN. This is achieved by reducing the spatial dimensions of the input ready for the next convolutional layer through the use of some function. Pooling only affects the width and height of the input, not the depth, as all depth channels are required to keep the colour mapping of the image intact.

Pooling as such inevitably leads to a reduction in the amount of information available to subsequent layers; this is advantageous however as it leads to less computational complexity, aiding in the minimisation of overfitting in the model.

A number of different pooling layer architectures exist in literature, such as max pooling, average pooling [20], and stochastic pooling [181], although more recent architectures favour the use of strided convolutions, rather than pooling, to reduce the size of the output feature map.

Fully Connected Layers

Fully connected layers take feature maps produced by the preceding convolutional and pooling layers and reduce these down to a single N-dimensional vector, where N represents the total number of classes and each dimension's value is the probability of the class. These probabilities are achieved using a softmax activation function, which takes the exponents of each input and normalises them by the sum of all inputs, giving values between 0 and 1. Outputted N-dimensional vectors can then be considered a feature map in their own right for further processing [78] or as a category for classification as the last layer of the network [55].

For example, let's assume we have a network who's aim is to classify an image into one of five classes. At the end of this network would be a fully connected layer which would take a feature map, transformed by the previous network layers, and produce as output five values between 0 and 1. Each of these values is required to be outputted by its own neuron, resulting in a fully connected layer with five neurons where each neuron represents a possible class for the image. The image's classification is provided by whichever neuron outputs the highest value.

Layer Architectures

Using the three layer types described above it is possible to create, in theory, an infinite number of CNN architectures all with different amounts and combinations of layers. There is no guarantee that every possible architecture will perform well however (indeed, one possible combination would be a single fully connected layer, which would not perform well at all).

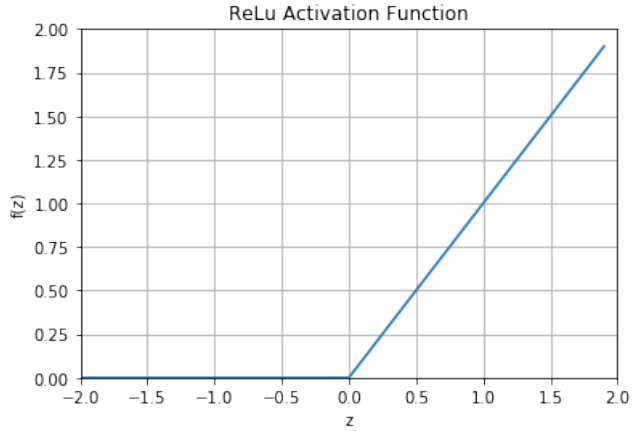

Figure 2.4 A visualisation of the ReLU Activation Function

Whilst it may be advantageous for certain areas of research to create their own custom CNN architecture, either through trial and error or the more recent approach of Neural Architecture Search [42], this is not applicable for most cases. For the vast majority of cases, there exists in literature well-defined generalised CNN architectures, and it is often these architectures which are utilised for computer vision tasks.

LeNet [82] was the first well defined CNN architecture. LeNet was only 7 layers deep, but performed well enough to be applied by some banks for automatic recognition of numbers on cheques. It wasn't until around 2012 that more attention was paid to these defined architectures however, thanks to AlexNet [78]. Utilising a similar but deeper architecture to LeNet, with more filters and a larger number of stacked convolutional layers, AlexNet also included now common CNN building blocks such as dropout [145], max pooling [20], and ReLU activation functions; the most popular non-linear activation function currently in deep learning, especially in computer vision [66].

The activation function is responsible for deciding which neuron in the layer passes it's value to the layer below by computing the weighted sum of the inputs and passing the result through a non-linear function. ReLU's non-linear function returns 0 for any negative value, or for any positive value x, it returns x. This can be written as $f(x) = max(0,x)$, and visualised in Figure 2.4. It is this non-linearity that allows for backpropagation to occur.

In 2014, Google introduced GoogleNet, also known as an Inception architecture, to the ILSVRC14 competition [147]. This net achieved a top-5 error rate of 6.67%, very close to what untrained humans could achieve on the competition dataset. This was achieved through a 22 layer deep CNN utilising several small convolutions, reducing the number of parameters from 60million in AlexNet to 4million in GoogleNet.

Finally ResNet was introduced a year later at ILSVRC15. This architecture can be as large as 152 layers deep, and achieved a human-beating top-5 error rate of 3.57% [65]. Shallower versions of ResNet exist, such as ResNet50 and ResNet101, which are 50 and 101 layers deep respectively.

2.3.2 Object Detection

As discussed, CNNs are now one of the main tools available for computer vision tasks. Object detection tasks are no exception, with CNNs now being utilised en masse to attempt to identify distinct regions containing task-specific classified objects in images and video. Whilst this is often performed in one of two ways, it is important to note that all object detection is still in essence a series of tasks performed via network layers.

Region Proposal Networks

The first, known as a Region Proposal Network (RPN), attempts to find image regions likely to contain objects of given classes. Training data is usually provided in the form of bounding boxes drawn around objects of interest and labelled with the corresponding class. These RPN detections can be relatively fast, using a selection search [156] gives around 2000 region proposals, known as RoIs, in only a few seconds on a CPU. Example RoIs can be seen in Figure 2.5.

Selection search is most commonly used with the R-CNN object detection algorithm [55]. This algorithm has a high recall rate due to the large amount of proposals, as there is a high probability that some of these proposals will contain Regions of Interest (RoIs) containing the objects being searched for. However, this can be time consuming and computationally expensive (although less computationally expensive than just sliding a window over the full image) as the network needs to be trained to classify these 2000 region proposals, taking up a large amount of disk space. Detection can also be slow using a vanilla R-CNN and, with the selection search being fixed, no adaptive learning takes place here which may lead to bad region proposals throughout.

Some of these time drawbacks were fixed in later versions of R-CNN, known as Fast-RCNN [54]. Rather than feeding the region proposals generated to the CNN, this algorithm instead feeds the input image to the CNN and generates a convolutional feature map. RoIs can then be taken from the feature map using selection search and warped into a shape suitable for the pooling layer, before being reshaped again into a fixed size for the fully connected layer. This is advantageous as it allows us to reuse some computations and allows

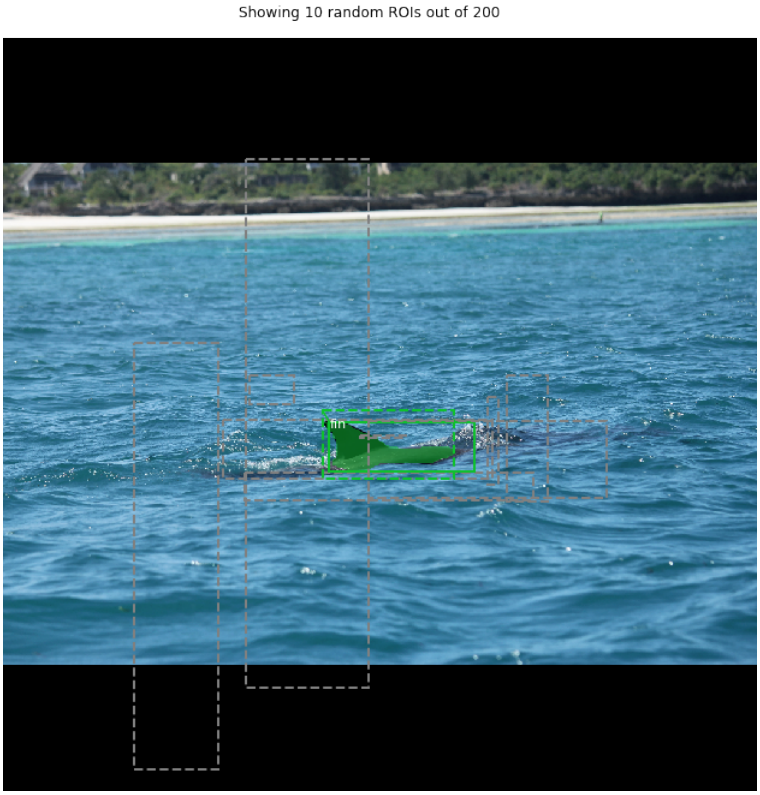

Figure 2.5 An example of RoIs generated on an image by an RPN, showing 10 random proposals out of the 200 generated. Note that two of the RoIs have been successfully classified as `fin`.

for backpropagation to occur throughout the network, greatly improving run times. This also means however that the runtime is dominated by how fast RoIs can be generated.

To fix this issue, Faster R-CNN was developed [129]. Now, instead of utilising selection search to generate the RoIs we can utilise a separate network to predict RoIs which can then be used to classify images within the regions. With this, we now train with four losses;

1. An object/not object classification from the RPN, 2. The RoI shift, 3. The object classification, 4. Final bounding box co-ordinates.

Detection Without Proposals

One issue with all RPNs is that they generally take a significant amount of time in order to classify objects in images, with the bottleneck being the region proposal generation. Because of this, there are algorithms which attempt to remove the region proposals altogether and instead look at the whole image. This input image is divided into an equal size grid. Within each square of the grid, we take a set number of bounding boxes which the CNN provides classification confidences for. Any above a set threshold are used to locate the object within the image. These algorithms are essentially one large CNN rather than splitting into a CNN and an RPN and are thus much faster although are not as accurate, especially on smaller objects due to the spatial constraints of the algorithm. Examples of detection without proposal systems include YOLO [126] and SSD [93].

2.3.3 Semantic Segmentation

Along with object detection, semantic segmentation is one of the key research areas in computer vision. Rather than providing bounding boxes around objects of interest as output, semantic segmenters aim to provide fine-grain categorisation for every pixel in an image, grouping pixels together in object classes known as masks. An example of an image and it's masks can be seen in Figure 2.6.

In general, semantic segmenters can be thought of as having two main components; an encoder, usually a pretrained classifier built with a standard detection architecture such as ResNet, and a decoder whose job is to project the coarse grain features learnt by the encoder to a fine-grain pixel space. There are two main ways to approach this decoding step.

The first is to use an RPN to perform region based semantic segmentation, extracting the regions from an image and then describing them. Each pixel of the image is then given a classification based on which highest scoring region it is contained in. Note that any pixels not in a region are given the class label of `background`. This is achieved through the use of a lightweight binary classifier ran over multiple proposal boxes, known as anchors,

Figure 2.6 An example of an image and it's corresponding ground truth `fin` masks.

covering the image at different scales. Each anchor is given an object score denoted by Intersection Over Union (IOU), a measure of how much overlap there is between a model's predicted bounding box and the ground truth. This is taken at a set confidence threshold, usually 50%, as the model will predict potentially hundreds of boxes for an image, all with different confidence levels. The vast majority of these predictions will be wrong, but will also (hopefully) have very low confidence scores and so they can be safely ignored and thus not counted in evaluation metrics. Taking a predicted bounding box B_p and a ground truth box B_g, the IOU between the two can be defined as:

$$IOU = \frac{\text{Area of overlap}(B_p, B_g)}{\text{Area of union}(B_p, B_g)} \tag{2.1}$$

Anchors with an $IOU >= 0.7$ with any ground truth are donated as positive anchors and are passed on for classification. Those with an $IOU > 0.3$ are considered negative anchors, and those where $0.3 <= IOU < 0.7$ are denoted as neutral anchors and are not used for training.

Usually however, positive anchors do not fully cover the ground truth object. Because of this, the RPN regresses a refinement applied to the anchors, shifting and resizing them to correct their encasement of the ground truth object. An example of this can be seen in Figure 2.8.

Utilising RPNs does have disadvantages however. Generating the segmentations from the regions take a significant amount of time, and the features generated by RPNs generally do not contain enough feature information to generate well defined masks. Recent research has attempted to fix these issues, such as SDS [61] or Facebook's Mask R-CNN [64].

Second, a Fully Convolutional Network (FCN) can be utilised for semantic segmentation [95]. An FCN learns pixel to pixel mappings without the need for region proposals, whilst

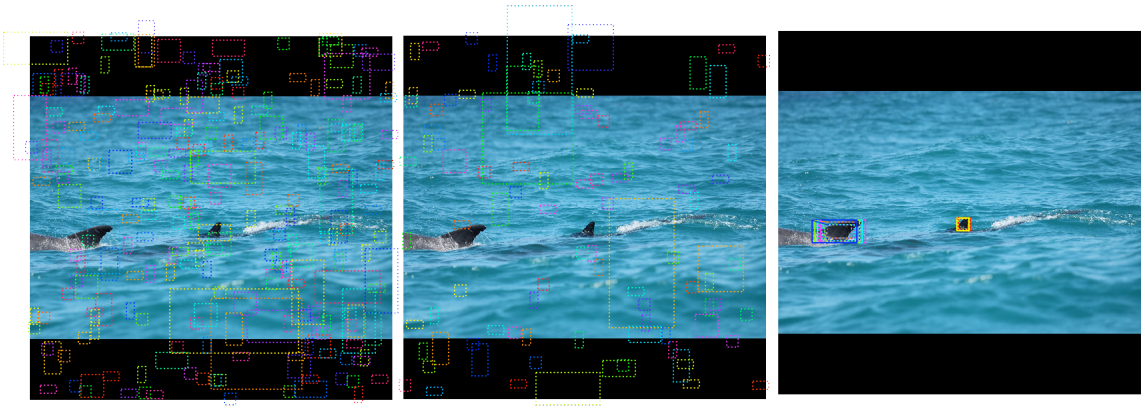

Figure 2.7 Generated anchors. Left: negative anchors. Middle: neutral anchors. Right: positive anchors.

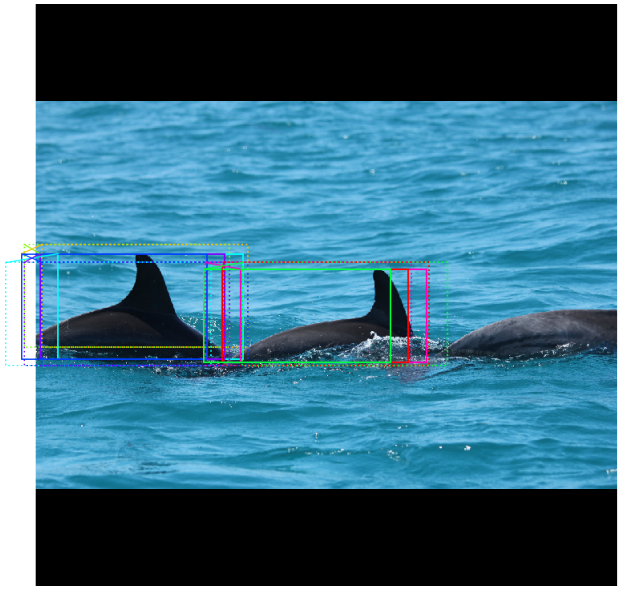

Figure 2.8 An example of refined anchors. Positive anchors before refinement are dotted, after refinement are solid.

also only including convolutional and pooling layers, allowing for an input image of arbitrary size (compared to classical CNNs which are generally constrained by a preset image size). This does lead to the disadvantage of down sampling the resolution of the outputted feature maps, leading to sometimes ill-defined segmented boundaries. This issue has attempted to be corrected however with more advanced FCNs such as SegNets [5] and DeepLab [28].

Semantic segmentation can be aided through forms of supervised learning. Providing training images which have been given pixel by pixel segmentation masks can greatly improve segmentation class accuracy. Creating these masks can be extremely time consuming for researchers, and is often farmed out to external companies such as Amazon's Mechanical Turk [21].

2.3.4 Part Segmentation

Whilst fine-grained visual categorisation is still an area of new research, an emerging approach to tackling this problem is through the use of part segmentation, whereby a coarse-grained classification is broken down into sub-components which are then analysed to provide a fine-grained identification [183]. This is still an active area of research, with some approaches focusing on a form of hierarchical part matching [178], some on alignment of objects to define a super-class shape [50], some utilising deformable part descriptors [184], and others using part localisation [91].

2.3.5 Instance Segmentation

Building on the concept of semantic segmentation, instance segmentation can be performed when further detail about an image is required by a developed system. Whilst many of the underlying processes are similar between the two segmentation types, instance segmentation allows for the model to distinguish between multiple objects which are of the same class; an example of this can be seen in Figure 2.9.

As such, instance segmentation provides a far more detailed explanation of the input image. This information can be invaluable if the developed system is required not only to understand what pixel classes are present in the input image, but also how many of these class instances there are. In a sense, instance segmentation can be seen as combining both object detection and semantic segmentation into one task. Traditionally however, in order to achieve the goal of instance segmentation, proposed systems have kept the two tasks divided. These *traditionalist* methods take one of two approaches.

The first, known as *top-down*, begins by detecting objects of interest via an RPN to create bounding boxes. These detections are then fed to the mask predictor to determine which

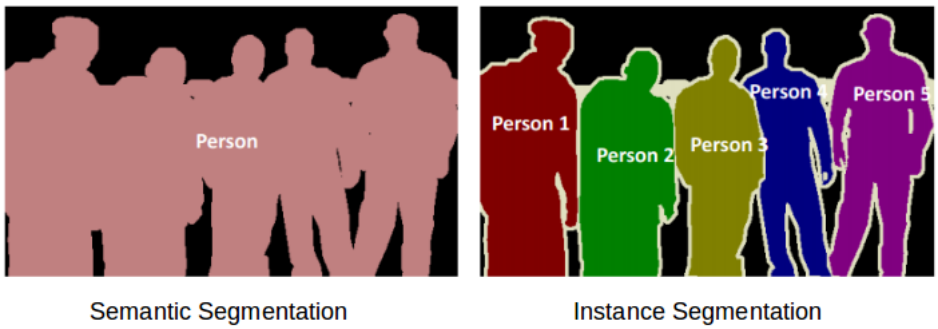

Figure 2.9 An example showing the difference between semantic and instance segmentation. In semantic segmentation, all pixels which belong to the `person` class have been classified as one `person` object. In instance segmentation, five `person` objects have been detected, and all `person` pixels have been assigned to one of the objects. Image from [141].

pixels inside of the box belong to either the target or background class. Examples of *top-down* approaches include Faster R-CNN [129], and Mask R-CNN [64]. In contrast, *bottom-up* systems first segment then detect, such as SpatialEmbedding [107] which attempts to tackle instance segmentation through the use of a Gaussian function to produce a probability for a pixel being part of the background or foreground, and then performing object detection on the foreground pixels. *Bottom-up* approaches often fail to reach similar levels of performance as *top-down* approaches however, and are thus rarely used. The major similarity between both *top-down* and *bottom-up* approaches is that they are both sequential in nature, requiring one stage to happen before the other. As such, these systems are very hard to speed up and are far from real-time. However, two stage systems often perform the best in terms of accuracy, and thus are still extremely common backbones of systems requiring the use of instance segmentation.

In recent years research into the development of real-time instance segmentation has shifted to utilising a one stage approach. These one stage systems are often able to achieve near real-time performance, although often do not produce levels of segmentation accuracy seen when utilising two stage systems. ExtremeNet [186] works to extract four "extreme points" and one "center point" of potential objects in the input image through the use of a keypoint estimation network, creating a coarse mask. ESE-Seg [179] utilises the concept of Chebyshev polynomials to fit a radius around each object inside of the detected bounding box. Similarly, PolarMask [177] also represents masks through the use of a contour around the object, modelling this through the use of polar coordinates. FourierNet [131] builds on this radius concept further through the use of a Fourier transform to smooth the contour. This contouring of the object is extremely fast, however the generated masks are very imprecise.

Further, any objects which contain spaces or holes, such as doughnuts, would not be able to be accurately represented.

YOLOACT [16] builds on the well known YOLO object detection architecture, specifically YOLOv3 [127], adding a branch for mask prediction, but performing this through the use of two parallel tasks. The first utilises an FCN to generate prototype masks, whilst the second predicts instance coefficients. These can be combined into one mask through matrix multiplication operations with the detected bounding box. BlendMask [27] works in a similar way to YOLOACT however predicts an attention map rather than instance coefficients and utilises FCOS [152] as a backbone, a completely anchor and proposal free object detection architecture resulting in reduced complexity when compared to YOLO [126] and SSD [93].

Whilst the majority of one stage approaches to instance segmentation rely on bounding boxes, this is not always the case. SOLO [166] introduces the concept of instance categories, assigning categories to each pixel according to the size and location of the instance. SOLOv2 [167] builds on SOLO through the implementation of an novel non-maximum suppression algorithm. SOLOv2 often depicts higher quality masks than more often used two-stage systems such as Mask R-CNN and is able to perform real-time inference, although it should be noted that both SOLO and SOLOv2 are extremely recent additions to the instance segmentation arsenal, both being released in 2020.

Mask R-CNN

As discussed in previous Sections, there are multiple standardised architectures utilised for segmentation tasks. As such, when developing a system which utilises segmentation developers of these systems will, more often than not, use one of the many architectures from literature rather than developing their own custom architecture. Utilising one of the standard architectures has many advantages; for one, researchers do not need to spend time creating a model architecture for their task, allowing for development in other, novel areas. Further to this, utilising a standard architecture allows for research to be more easily understood and reproduced. As this thesis focusses on the automation of photo-identification systems rather than on the development of new novel architectures, it makes sense to make use of an architecture which is well known, has a track record of performing well when trained on non-benchmark or custom datasets, and is easily reproducible. As such, parts of this project's automation pipeline make use of Mask R-CNN [64]. Because of this, it is important to understand Mask R-CNN in more detail compared to the other architectures discussed previously in this Chapter.

As we have seen previously, it is often the case that new architectures either extend or borrow features from older ones. This is also the case with Mask R-CNN. Developed in 2017

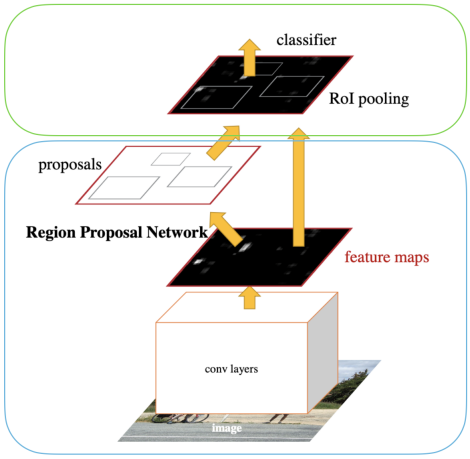

Figure 2.10 The Faster R-CNN architecture [129]. The blue box represents operations in stage one. The green box represents operations in stage two.

by He *et al.* at Facebook AI Research, Mask R-CNN was developed on top of the existing 2016 Faster R-CNN architecture from Ren *et al.* [129] (itself an extension of Fast R-CNN developed in 2015 [54]).

Faster R-CNN is a two stage architecture. The first stage utilises a standard backbone network, typically ResNet [65], VGG [143], or Inception [147], to convert an input image into a set of feature maps which are passed to an RPN for analysis (see Section 2.3.2 for a breakdown of RPNs). This RPN generates region proposals which are passed to the second stage of Faster R-CNN, along with the previously generated feature maps, and fed to an RoI pooling layer. Here, each proposed region and corresponding feature map is utilised to predict bounding boxes, classifications, and confidence scores. A visual representation of Faster R-CNN's architecture can be seen in Figure 2.10.

Mask R-CNN extends Faster R-CNN, allowing for instance segmentation through some relatively simple changes and additions to stage two of the architecture. First, the RoI pooling layer is replaced with an RoI align layer. This replacement layer removes the "harsh quantisation" which is present in RoI pooling, and properly aligns the extracted features with the input image. Second, an additional branch is added to the end of stage two. This branch receives the output of the new RoI align layer and processes it using a *mask head*, consisting of additional convolutional layers which generate pixel predictions and instance mask outputs. See Figure 2.11 for a visual representation of the changes made by Mask R-CNN.

Thanks to these additions, Mask R-CNN is able to perform extremely accurate instance segmentation with a relatively small drop in inference speed. Whilst it is not real-time, this

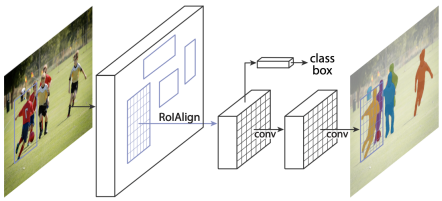

Figure 2.11 A visual representation of the changes made to Faster R-CNN to create Mask R-CNN. [64]

is an acceptable trade off for the accuracy of predictions on custom datasets. Indeed the use of Mask R-CNN for instance segmentation in literature is far ranging, being utilised in medical [3, 30, 92, 132], agriculture [32, 84, 122, 185], sports [22, 108, 119], astronomy [23], and nautical [70, 109] fields. As well as being well known, Mask R-CNN is also extremely reproducible. An official PyTorch implementation is available through Facebook AI Research's Github [174], whilst Matterport's Mask R-CNN implementation is most commonly utilised when working with Tensorflow (including in this project) [165].

2.3.6 Fine-Grained Visual Categorisation

Categorisation of objects through the use of machine learning may at first glance look like a solved problem. Indeed, it is now possible to achieve extremely high levels of accuracy on a wide variety of tasks; at the time of writing the current state of the art for ImageNet [114] and CIFAR-10 [77], two of the most commonly used classification benchmark datasets, are both held by Foret *et al.* utilising EfficinetNet with SAM [46] at 88.61% and 99.70% top-1 accuracy respectively. Even outside of these benchmarks however, high accuracies would be expected for more complex use cases where custom datasets have been created, such as in manufacturing environments. However, it is important to note that all of these tasks are coarse-grain in nature. Benchmark datasets usually contain classes which are relatively distinct, for example `cat`, `dog`, and `ship` classes in CIFAR-10, which all have large interclass variation.

In more recent years the focus of research in object detection has thus mostly shifted from the coarse-grain to the fine-grain domain, where distinct classes have small interclass variation. For example whilst CIFAR-10 has one class covering all different types of `dog`, the fine-grain dataset Stanford Dogs [75] is made up of 120 classes each containing examples of only one dog breed each (`chihuahua`, `beagle`, etc.). Other common fine-grain benchmark datasets often focus on wildlife or vehicles, including Caltech-UCSD Birds 200 [171] and

the updated Caltech-UCSD Birds 200-2011 [164], iWildcam for camera trap data [8], and FGVC Aircraft [99].

Whilst fine-grain datasets may contain small inter-class variation, their intra-class variation is often large. Class examples often contain a wide variety of orientations, poses, colour, and sizes. This allows for trained models to be more generalisable and capable of detecting class examples in previously unseen images where the example may be obscured or unlike conventional class examples. It is also important to note here that models which perform well on coarse-grain data are not guaranteed to do so on fine-grain data. For example EfficientNet with SAM which, as previously stated, is state of the art in multiple coarse-grain tasks currently ranks 31st in the FGVC Aircraft benchmark ranking on Papers With Code [1].

2.4 Computer Vision for Conservation Technology

Thanks to the large advancements in computer vision and deep learning, and the increasing prevalence of these systems in areas such as manufacturing and healthcare, researchers have, in recent years, began exploring other areas of society which could benefit from AI systems. One of the more niche, but arguably highly important areas where computer vision can make an impact, is conservation [170].

Work in conservation tech has been mostly dominated by machine learning systems for camera traps. This makes sense, as this data collection technique usually operates over vast spatial and temporal areas. As such, a large amount of varied data already exists for systems to be trained on. For example, the Snapshot Serengeti project [2], developed by Swanson *et al.* has utilised camera traps in Tanzania's Serengeti National Park to develop a fully labelled camera trap image dataset capable of training machine learning systems. The camera traps used have been in continuous operation since 2010 and cover an area of 1125km^2 [146]. The iWildcam dataset provides further camera trap training data from across the South-western United States [8].

A second advantage of camera trap data is their high false positive rate. These traps capture a photo every time movement in the frame is detected, and as such a large proportion of the images a camera trap captures either do not contain any animals at all (e.g. wind has caused the surrounding vegetation to move), or contain animals which are not the primary species of investigation. This large false positive rate provides a key driver for the development of machine learning camera trap systems which could, for example, filter out

[1] Papers With Code - FGVC Aircraft Rankings: paperswithcode.com/sota/fine-grained-image-classification-on-fgvc

[2] Snapshot Serengeti: snapshotserengeti.org

these false positive captures automatically. Whilst these images may simply be discarded by the researchers, they have a use in the development of machine learning camera trap based systems, allowing the system to be trained on a wide variety of false positive examples. As such, machine learning systems developed for camera traps have found quick adoption in the conservation community with many systems now capable of performing fine-grained species classification with extremely high accuracy [9, 111, 112, 148, 173]. Recent work by Clapham *et al.* has moved further to the extreme of fine-grained classification with BearID, a project which adapts human facial recognition systems for use with brown bears (*Ursus arctos*) via metric embeddings, achieving an "individual classification accuracy" of 83.9%. Here, BearID is not classifying the species *Ursus arctos* but rather individuals within the *Ursus arctos* population, a significant achievement given the challenge of identifying individuals within a species which do not have unique markings [136].

Marine conservation researchers, in contrast to their on-land counterparts, have been slower at adopting computer vision to aid their research. This is, in part, due to the relative lack of available data to train systems. Due to the unsuitability of marine environments for the deployment of camera traps, conservationists here traditionally rely on identification from photographs taken either from the coastline or from a vessel. As this requires a human operator, the size of datasets available is relatively small. Furthermore, given the high cost of data collection, marine conservation groups often keep a tight grip on their data. This has led to a lack of available open-source datasets for those who wish to train machine learning systems for use in marine conservation. Thanks to advances in UAV technology and their current inexpensiveness, some research groups have shifted focus to the use of drones for image capture. This new approach has seen success in areas such as photo-identification [15, 56], microbial sampling [25], and human-interaction response monitoring [44]. The use of drones for this time of work is not yet mainstream however, and some recently published work highlights the need to better understand how drones affect the behaviour and health of marine species [12, 52, 120, 124].

2.4.1 Utilising Computer Vision in Cetacean Conservation

The idea of utilising statistical methodology and machine learning in a marine cetacean space has, in recent years, been gaining popularity, with multiple papers being published in this area. Karnowski *et al.* propose using Robust PCA to subtract background from underwater images to help identify captive bottlenose dolphins, and track their movements through multiple distinct areas, allowing researchers to annotate pool positions 14 times faster than before [74]. Bouma *et al.* provide a system focusing on metric embedding learning to photo-id individual common dolphins, achieving top-5 accuracy scores of around 93% [19]. Further, Quiñonez *et*

al. propose a CNN based system to detect four classes: `dolphin`, `dolphin_pod`, `open_sea`, and `seabirds` [123].

Outside of statistical theory and academic papers, multiple systems have been developed, or are currently in development, to aid cetacean researchers in the photo-id process. The first of these, known as FinScan, was developed in 2000 [68]. This is a semi-automated photo-id assistant whereby the user imports images taken during fieldwork which FinScan then attempts to create a trace of the fin in the image. Users may manually edit this trace however if it is not exact (this feature was developed due to frustration with barnacles attaching to fins in the area where FinScan was developed). This trace of the fin is then checked against a local Microsoft Access database to determine close matches which are presented to the user. Before images are imported into FinScan they must be manually cropped, sharpened, and rotated by the user. Rotation of the image is especially important, and the FinScan algorithm is not rotation invariant. Further to this due to FinScan being an old piece of software, whilst it is freeware it is no longer readily available. Anyone who wishes to use it must procure a copy from someone else, there is no central repository for downloading. Issues with running the software on newer systems may also be present.

Similarly to FinScan, FinBase is a photo-id database management system developed by NOAA Fisheries [45]. However, unlike other systems, FinBase provides no matching based on automatically generated fin properties; instead, FinBase facilitates matching through user defined attributes. These could be physical descriptors such as 'top notch' or 'skin disorder' but may also be non-physical attributes if the user wishes. Fins are partially matched based on querying the backend database for entries which also have the attributes of those inputted by the user for the query fin.

Alongside both of these, DARWIN [58] provides automated identification of new images based on those already in the attached database. Like FinScan, users of DARWIN trace around the leading and trailing edges of the fin they wish to identify. These edges are stored in a database as a set of evenly spaced points approximating the outline of the fin which is then used for identification.

Photo-ID Ninja[3] is a system currently in development which aims to photo-id individuals based on pigmentation commonly found on fins of the New Zealand common dolphin (*Delphinus* spp.) [53]. Here, pigmentation is used due to the low chance of other prominent markings becoming present on individuals. Matching is performed via the Euclidean distance between the input image and the catalogue of known individuals, which is then sorted by ID and validated using 5 fold cross-validation [18]. Current reported accuracies for this pigmentation matching are a top-1 accuracy of 90.6%, top-5 accuracy of 93.6%, and an mAP

[3]Photo-ID Ninja: photoid.ninja

of 80.8%. This pigmentation matching is still in development however, although Photo-ID Ninja does currently provide a bulk cropping mechanism to aid in the speed up of manual identification. Users can provide a batch of images taken directly from the field to the system, which then outputs a zip file of cropped fins which can then be manually identified.

Work undertaken by Georgetown University and Google in the area of cetacean photo-id has also provided promising results [86, 100, 157]. The system, which utilises Google's Cloud Auto ML framework, can quickly identify bottlenose dolphins from Australia's Shark Bay. This system shows users the top-200 closest matches along with their confidence scores utilising both the leading and trailing edges of the fin. It is reported that this system saves Georgetown University's cetacean team around 4500 hours per year, highlighting the need for systems such as these to researchers in this field. However, this system does not link to a backend database to log matches found - this must be done manually by the researchers. Further, any new individuals which need to be added to the system, or indeed if the system was to be redeployed to a new area, then all training of the underlying model must be performed by Google engineers rather than locally by the researchers who wish to utilise the system. Further, fins to be identified must be inputted one by one, no batch input function exists.

In the past few years, systems utilising CNNs have started to enter the photo-id space. HappyWhale[4] is a CNN based photo-id system focussing on humpback whales (*Megaptera novaeangliae*). The underlying CNN for this system was developed through a Kaggle competition[5] by user Jinmo Park to identify patterns present on the tailstock of the humpbacks [73], utilising elements of ArcFace [38] and DeepFace [149] to do so. Users interact with HappyWhale through their website, uploading images of the tailstocks encountered. The HappyWhale system then attempts to identify the individual before presenting back to the user. If the user provides location data, HappyWhale also keeps track of this to produce travel maps for the individuals, as humpback whales are known to travel vast distances in their lives. Success rate for HappyWhale varies greatly, from 99% for "good to high quality" images to 50% for full fins at 50x50px. HappyWhale struggles with partially obscured tailstocks however, and work is currently ongoing in this area [**?**].

Of all the systems in use or development today, one of the largest and most well known is FlukeBook[6], a fully automatic photo-id system for bottlenose dolphins, humpback whales, and sperm whales (*Physeter macrocephalus*). This system is part of a wider network of animal identification tools based on Wildbook, an open source software framework developed by non-profit organisation WildMe to facilitate the introduction of artificial intelligence into

[4]HappyWhale: happywhale.com
[5]Kaggle competition: kaggle.com
[6]FlukeBook: flukebook.org

the conservation space [11]. Within FlukeBook there are two main photo-id algorithms; CurvRank and FinFindR.

CurvRank is an algorithm developed by Weideman *et al.* [169] which automatically identifies the trailing edge of the fin and represents this as a set of ordered coordinates. Each coordinate point then has a circle of radius r placed upon it, before being transformed horizontally. The curvature at this point for a given r value is then defined as the ratio of the area under the curve againsy the area of a square around the curve of length $2r$. This allows for the definition of the trailing edge of the fin to be rotation invariant.

FinFindR is an algorithm developed by Thompson *et al.* which allows for inputted images containing bottlenose dolphins to be identified. FinFindR works with uncropped images of the whole area unlike other systems which require just the fin to be in the image. FinFindR then automatically detects the dolphin and crops it out, saving a new image. Cropping can be performed on either the whole body or on the dorsal fin only. This cropped image is then passed to a canny edge detector which creates an embedding of the trailing edge of the fin. This embedding is mapped into a high dimensional space based on work in FaceNet [140], with clustering of individuals achieved using Ward's variance minimising clustering [168]. Reported accuracies for FinFindR currently stand at a top-1 accuracy of 88%, top-5 accuracy of 94%, and top-50 accuracy of 97% [151]. It should be noted however that FinFindR has currently only been tested on bottlenose dolphins and work is still ongoing. The code for FinFindR can be found on GitHub [7].

Recent work undertaken by Lee *et al.* proposes a novel algorithm for cetacean identification [85]. The proposed system is capable of detecting small objects in large images, and utilises this for fin detection. Next, segmentation is performed using U-Net [133]. The resultant output is then passed to a post processor which re-aligning and normalising the fin. The most significant features of the fin are then extracted and passed to a VGG based system [143], combined with a novel V2BC component, for identification.

Maglietta *et al.* propose DolFin [98], a SURF based system [7] building upon work undertaken by Renò *et al.* [130] for identifying Risso's dolphins. As mentioned in Section 2.1, this species is susceptible to prominent long-term scarring, and thus is well suited to feature detection algorithms such as SURF, with published results showing a much greater identification accuracy can be achieved compared to utilising common photo-id aides such as DARWIN [58].

Morteo *et al.* [106] perform semi-automatic fin measuring using FinShape in order to aid in population monitoring. This technique, based on [172], does not require the user to trace

[7] FinFindR: github.com/haimeh/finFindR

the fin; instead lines are projected out from the base of the leading edge of the fin, with the user cutting these lines where they intersect with a point on the trailing edge.

2.5 Summary

This Chapter presents the key ideas required to understand and appreciate the novel work proposed in this thesis. An extensive overview of deep learning and computer vision is provided, as well as an introduction to photo-id, a key methodology for mark-recapture surveys utilised by conservationists. A summary of previous research combining conservation and deep learning has also been provided, with a focus on marine environments and cetaceans.

The ability to perform object detection on photo-id fieldwork data to identify regions in images where cetaceans are present is a key component of this thesis. Chapter 3 will discuss this work in more detail, including an in-depth analysis of the cetacean detector's requirements, as well as its implementation and evaluation.

Chapter 3

Cetacean Detection Using Deep Learning

When building any large-scale project, it is important to break the task down into various subcomponents. In this Chapter we will examine one such subcomponent utilised in the developed automatic photo-id system, the cetacean detector. This component takes images captured during photo-id surveys and detects regions of interest, in our case these are dorsal fins which have breached the waterline. This Chapter will discuss the requirements a detector must meet, how it was trained, how the optimal hyperparameters were found, and how the detector can be utilised to provide a downstream identification system with only the information it needs.

3.1 Requirements of a Cetacean Detector

Before a system for automatic cetacean detection can be developed, it is important to first define the problem and understand the requirements of the system. The overall aim of the detector is to be able to take large-scale images as input, fed in one at a time, and process them in order to locate regions of interest. Unlike other automated detection systems common in literature, this system is only required to detect one class of object, `dolphin`. These detected regions can then be passed further down the system pipeline to the identifier.

As such, this detector can be considered a coarse-grain task, and at first glance may seem somewhat trivial. However, due to both the nature of the environment in which the class must be detected, and the technical requirements the system must perform under, this becomes a much more complex problem.

Figure 3.1 Some cetaceans, such as bottlenose dolphins, travel in pods. The developed detection system must be capable of splitting this pod into individual animals to be passed to the identifier.

3.1.1 Environmental Requirements

First the area in which this system is to be deployed, in open water, is susceptible to adverse weather conditions such as high winds. This in turn leads to sub-optimal conditions for detection which the system must be capable of handling, most notably high amounts of sea swell. Further to this, dolphins, the main data subject of this work, are communal and travel in pods. An example of this behaviour can be seen in Figure 3.1 Thus, the system must be capable of differentiating between overlapping individuals. Even if not all of the overlapping individuals are suitable for identification down the line, the system must still be able to separate them into individual detections to prevent further misclassification downstream; for example, lets assume an image is inputted to the detector containing three overlapping individuals where one of these has markings clear enough for a human to identify the individual. For our system to also be able to do this, the detector must be able to separate the overlapping animals into the individual components. This allows for the identifier to be provided with three images, one of which is the individual that is identifiable. If all overlapping individuals were passed to the identifier as one, the chance of identification will be greatly reduced, and the image may be identified as `noise`.

Next, the detector must be capable of differentiating between dolphin fins and waves. Again this might sound trivial, but thousands of years of evolution have resulted in fins and waves looking extremely similar to the untrained eye, which artificial ones often start out as. Especially from a distance and in choppy waters, fins and waves often have extremely similar

Figure 3.2 Two images of the same individual taken from different angles of approach, directions of travel, and distances from the vessel. Note how this changes the make-up of the dorsal fin, however keeps the identifying notch visible.

shape and dimensions. Furthermore, the animal's bodies are also similarly coloured to their surroundings. These adaptations allow the animals to be better protected and camouflaged in their environment, but can cause issues with detection systems. This becomes apparent when thinking about how CNNs *see*. As described in Chapter 2.3, CNNs see input images as a matrix of pixel values. When training an object detection system, the CNN is also told which parts of this matrix are related to a class, any without a class label are considered background. If fins and areas of background contain similar pixel values, and these pixel values are clustered in the same ways, this can result in issues when training a model to detect instances of a class without misclassifying the background.

Another important requirement is for the detector to be able to handle objects of varying size, shape, direction, and angle of approach to the camera. When working in an open water environment with live animals, the conditions that you will capture images of the animal under are extremely variable. You may capture an image of the animal at any point in it's breaching process, which will change the size of the dorsal fin which is above the waterline. As dorsal shape is unique to each individual, the detector must be capable of understanding a general 'fin-shape' rather than a fixed one.

Furthermore how the animals breach the waterline is also extremely variable. Breachings may occur at any direction relative to the boat and the animal could itself be travelling in a different cardinality. The ideal scenario in this case would be for a breaching to occur either directly East or West of the boat (off the port or starboard side respectively) and for the animal to be travelling perpendicular as this provides the best chance of mark capture, however this rarely occurs. For example, a breaching may occur off the port-side of the bow (approx North West relative to the boat), but the animal may be travelling in a South-Easterly

direction. These approaches greatly change the look of the fin, however they may still contain identifiable markings, thus our detector should be able to detect these fins and pass them along for identification. An example of this can be seen in Figure 3.2, which also shows how distance from the vessel can change the camera's view of the dorsal.

As mentioned previously, weather conditions can also greatly affect how a dorsal fin is captured by a camera. However, especially in mark-recapture surveys, there are only two conditions that need to be worried about; swell and lighting, other conditions do not need to be handled by this detector. This is due to most research groups having limits of sea state for safety reasons. With regards to Newcastle University's Marine MEGAfauna Lab, this limit is a sea state less than 3 on the Beaufort scale [144]. As such a mild amount of swell and splash can be expected, which the detector should be capable of handling. Lighting conditions are not considered in the Beaufort scale, but for operational reasons the vast majority of mark-capture surveys take place during daylight hours. This can lead to large amounts of glare in images, especially on clear days. As such, the detector should be invariant to these conditions.

3.1.2 Technical Requirements

As well as being able to handle a variety of environmental factors, there are also some technical requirements that the detector must meet. With all deep learning based computer vision approaches, there is often a trade off that must be made between speed and accuracy. In most cases, these are inversely proportional to each other; the faster a system is required to perform, the lower an accuracy you must be willing to tolerate - Huang *et al.* discuss this in greater detail in their 2017 paper [71]. Thanks to the pace of research in this area, 2020 saw the release of object detection architectures which can perform operations in real-time such as EfficientDet [150] and YOLOv4 [14]. Current results on benchmark datasets using these real-time architectures are still a long way off their non-real-time competitors however, and accuracies would drop further on custom non-benchmark tasks such as cetacean detection.

Because this trade off must be made, before deploying a deep learning model it is important to decide where the system will be utilised. As photo-id surveys are performed on small vessels such as RIBs, space is severely limited on board. Because of this, it is not appropriate to add additional hardware to the vessel to perform this analysis during the survey. Furthermore, the current methodology of cetacean researchers is to perform identification once back on land, even when utilising photo-id aides. As the system proposed in this project is intended to fit into existing procedures rather than replace researchers, it is appropriate for the system to also be land based rather than on the vessel. Thinking about the current procedure further, this project's proposed system could be, for example, left running

overnight performing identifications whilst the researchers are away or during the day whilst they are on surveys, being left to work through the last day's worth of images captured. As such, there is no need for the system to operate in real time to fit in with the current workflow of cetacean researchers, provided the system completes its task within a reasonable time frame. Further to this, as the output of the detection model will be passed to an identification module, it is imperative that as much noise is removed as possible during the detection. In order to do this, the accuracy of the detection must be as high as possible, furthering the case for an accurate system over a fast one.

This idea of reducing as much noise as possible can be used to further narrow down the requirements of the detection system. As discussed in Section 2.3, the output of detection systems can be provided in different formats. In bounding box detection systems the detected objects are denoted by a set of at least two pixel coordinates denoting the top-left and bottom-right extremes of the object. These detections are often more cost-effective, both from a labelling perspective requiring less person-hours to complete, and to perform computationally. However, a model trained on bounding boxes will produce outputs which will only partially remove noise deemed to be outside of the detection's extreme points.

If we utilise pixel wise mappings however, then each pixel is given a classification. This allows the system to be more discrete with its detection, telling us exactly where the object is in the image. These pixel locations can then be used to removed all background. Pixel wise labelling is far more labour intensive however, and thus more financially costly to produce. Both semantic and instance segmentation methods will allow the detector to utilise pixel-wise mappings to remove background noise. However, utilising our requirement defined earlier in Section 3.1.1 that the detector must be capable of reducing an overlapping pod to its individual component animals, requiring the use of instance segmentation in any pixel-wise detections would be preferred.

As such, this requirement reveals a further trade-off the system must make. The amount of noise removed by the detector is proportional to the cost and labour needed to create data to train the system; this was required to be explored in more detail before it could be decided if bounding boxes or instance segmentation masks would be more appropriate for use with the detector. This is discussed in more detail in Section 3.2.1.

3.2 Deciding on Architecture and Framework

Based on the requirements outlined in Section 3.1, it is possible to begin deciding on how the cetacean detector is to be developed. One of the most important factors in the overall approach taken in the detector's development would be its use of either bounding boxes

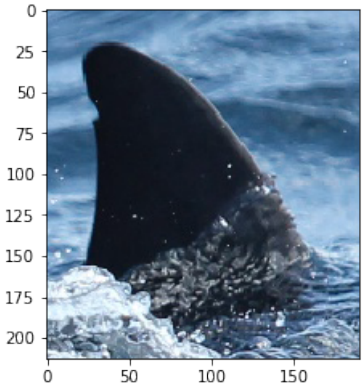

Figure 3.3 An example manual crop utilised in bounding box suitability testing.

or pixel-wise mappings during training. As mentioned previously, the use of pixel-wise mappings would allow for a greater removal of background noise, but is extremely costly and labour intensive to produce. In contrast, bounding box labels are easier and cheaper to produce but will lead to less background noise removal.

3.2.1 An Investigation into Bounding Boxes

Due to their relative cheapness and ease to produce, the use of bounding boxes in this project would be extremely beneficial. However, if the use of bounding boxes at this stage would hinder the accuracy of individual identification downstream, then this would outweigh the cost of pixel-wise mappings.

As such, an investigation was undertaken to decide whether bounding boxes would be a viable option, and if their use would hinder downstream identification. To begin, a small amount of data was provided by the Marine MEGAfauna Lab, discussed in more detail in Section 3.3.1, which contained images captured during a previous cetacean survey. A subset of this data was manually cropped to simulate the output of a bounding box detector, an example of which can be seen in Figure 3.3. This manually cropped data included some background but ensured the region of interest, the dorsal fin, was centred and prominent, representing an optimal output from a bounding box detector.

Feature Extraction with SIFT

To begin, processing of the cropped images focussed on the use of feature extractors such as Scale-Invariant Feature Transform, also known as SIFT [97]. As the name suggests, SIFT is invariant to scale, a major advantage for use with cetacean survey data where the region

of interest's size may change depending on when the image of the dorsal fin breaching the water is captured. If SIFT was capable of producing feature descriptors of the dorsal fins with only partial background removal, this would show potential for individual identification where some background is present, possibly through the use of the feature descriptors.

First SIFT was performed on the entire cropped image, however this proved unfruitful, picking out very few features in areas of the image which contained the animal's dorsal fin and instead focussing on the feature heavy areas present in the sea, even in images containing relatively calm water. This result indicated that further refinement was required, potentially reducing the area SIFT was allowed to explore.

Reduction of the search space available to SIFT was achieved through the use of colour thresholding. Here, a mask was created programmatically for each image based on bounded RGB colour values found in the dorsal fins, giving an upper threshold of (14, 16, 26) and a lower threshold of (54, 51, 66). As such, SIFT would only be performed in areas of the image where pixel values fell within this range. An example result of SIFT after colour thresholding can be see in Figure 3.4, with coloured circles surrounding an extracted feature. As can be seen, colour thresholding helps in removing a large amount of background water from the computation. However, issues arise where areas of water are also within this thresholding. Because of this, colour thresholding before SIFT only reduces the amount of features extracted from the water, it does not remove them, which may result in misidentification downstream.

Further to this, it can be seen that SIFT is incapable of extracting relevant features used in the photo-id process. For example, in Figure 3.3, a notch is clearly present on the dorsal fin which would be a good marker for individual identification. However, when performing SIFT on this dorsal as seen in Figure 3.4, note how this feature has not been detected by SIFT, which has instead detected an area above the notch which contains no identifiable information.

Feature extraction methods such as SIFT are also incapable of extracting other identifiable markers such as fin shape. As such, the use of SIFT was deemed improper for this use case. It is important to note here that the use of SIFT may be appropriate for cetacean species other than this project's data subjects of bottlenose and white-beaked dolphins. For example, the use of SIFT has been shown to be appropriate to aid in identification of individual Risso's dolphins [130].

Background Removal with GrabCut

Testing the suitability of SIFT as described in Section 3.2.1 highlighted the need for complete background removal before identification would be possible with bottlenose and white-

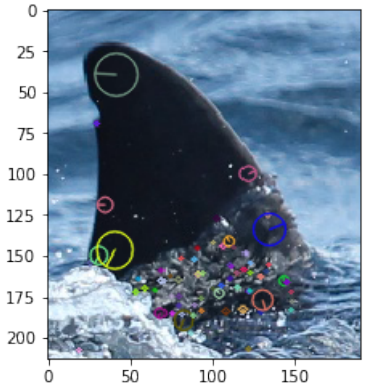

Figure 3.4 An example manual crop showing the result of SIFT feature extraction when thresholded based on RGB colour values.

beaked dolphins. In order for bounding boxes to be a viable option in this scenario, a robust background removal process would need to be created. Further, the process would need to be capable of operating under unseen conditions in an unsupervised manner without labelled segmented data to train on. If the background removal process required training data to operate, this would increase the overall cost and labour required to use bounding boxes, and as such reduces the suitability of them compared to utilising pixel-wise mappings from the beginning.

The current state of the art in this area is GrabCut, proposed by Microsoft Research [135]. This algorithm allows for the segmentation of foreground objects from the background with minimal or no human interaction. As GrabCut would be utilised in a fully automated setting, GrabCut would be required to perform background removal with no human interaction. Testing of the suitability for GrabCut was performed using the same cropped images as those used for SIFT testing. Again, issues arose when performing GrabCut on the cropped image data. The algorithm struggled to understand which parts of the image were classed as background and foreground, which resulted in imperfect segmentations. This was especially an issue where the dorsal fin was present in rough water, where splash would be in-front of the dorsal fin when captured by the camera. The use of GrabCut on Figure 3.3 can be seen in Figure 3.5.

As can be seen, the use of GrabCut as a background removal tool does not perform as expected on data the detector is required to operate on. Because of this, as well as the unsuitability of feature extraction as seen in Section 3.2.1, the use of bounding boxes in the cetacean detector stage was deemed not to be possible. As such, the focus of testing moved to the use of pixel-wise mappings and instance segmentation.

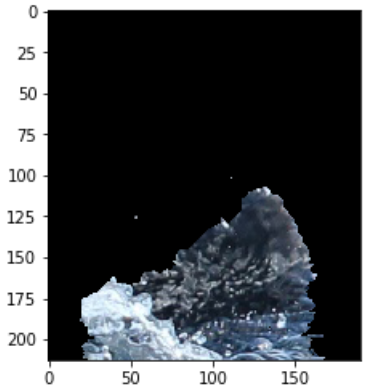

Figure 3.5 An example manual crop showing the result of GrabCut background removal.

3.2.2 Instance Segmentation Architectures

One of the major decisions that must be made here is which model architecture should be utilised in order to provide the required pixel-level detections. As this project is devoted to improving existing procedures and introducing deep learning to a novel space, it is far more advantageous to utilise existing model architectures rather than develop a custom one. The development of a custom architecture for this stage of the project would be extremely time consuming, taking away time from more novel parts of the project (notably the identification of the individual animals). Further, as this project is introducing deep learning methods to a novel space, the project needs to be able to convince researchers in this space that the system is reliable; this is more easily achieved using a pre-existing architecture where use cases already exist in literature and business.

To this end there are two main model architectures that can be chosen for this task; U-Net [133] and Mask-RCNN [64]. Both of these architectures work in different ways. Vuola *et al.* provide a more detailed comparison between the two models [163], however the main focus for this project is their resultant output mask structure.

U-Net is based on an encoder-decoder architecture. This allows for fast and simple segmentation when working with images where you only require one output. For example, taking U-Net's original use case of biomedical imaging, let's assume we have an image showing a group of cells and we wish to segment them into the individual components. U-Net is able to perform this operation efficiently through boundary estimation to locate the outer edges of the cells which allows them to be segmented from each other. However this results in an output of the same dimensions as the input, that is, all segmentations are provided in a single binary mask.

In contrast, Mask-RCNN utilises a multi-stage architecture (described in more detail in Section 2.3.5). This allows the architecture to place each detection on its own binary output mask. This is extremely important for our use case; as our detector will be used as part of a larger system, passing information downstream to the identifier which will require individuals to be in their own images, processing each individual in series. If U-Net was utilised for the detection stage, whilst initially being more efficient than Mask-RCNN, further processing of the binary output mask would be required to split this into it's individual components. In contrast, if Mask-RCNN was utilised then the processing required in between the detection and identification stage would be far simpler. Again, this allows for more time to be spent working on the novel aspects of this project whilst keeping the pipeline as simple as possible. This reason was a big factor in deciding to focus on Mask-RCNN for this stage.

Another factor which must be decided upon when starting developing a deep learning system is the language and framework to be used for development. With regards to language this was a fairly simple decision; the vast majority of deep learning research and development is written in Python. The language benefits from an efficient and lightweight syntax as well as having a host of different deep learning packages available to aid in development. Further to this, both of the major deep learning frameworks, Google's Tensorflow [1] and Facebook's Torch (of which PyTorch is the most actively developed) [116], both provide full Python support and have active communities for the language. Thanks to this, the vast majority of deep learning development is performed using Python in one of these two frameworks. By utilising these technologies, this project's code is easily reproducible and understood, as well as extendable in the future.

Of the two main frameworks, the use of Tensorflow was decided for this project. Whilst this decision was made somewhat due to personal preference, Tensorflow was (at least at the time of starting this project) the primary framework for development of deep learning systems outside of academia. Rather than developing a custom Mask-RCNN in Tensorflow for this project, Matterport's Mask R-CNN implementation [165] has been adapted. Whilst this does require the use of an old version of Tensorflow (1.14), this version is also stable and has a large library of available support unlike the most recent versions (>= 2.0).

3.3 Initial Testing of Mask-RCNN

In order to build a Mask-RCNN detector which fulfilled the requirements as laid out in Section 3.1 an understanding of the framework needed to be achieved. Thankfully, the downloaded repository also includes some tutorial notebooks, most notably an example on balloon segmentation which proved invaluable for learning the basics of how Mask-RCNN

operates both on a fundamental code level and at a higher level, understanding how the code can be adapted for other use-cases. In order to progress onto cetacean detection however, a dataset of cetaceans would be needed.

Exploration of available open-source datasets to find cetaceans in conditions this detector would be operating in proved unfruitful. Many standard benchmarking datasets contain animal classes, and thus an exploration of these was conducted. Of the more generalised benchmark datasets, those such as ImageNet [37] which contain a large corpus of varied classes, only CIFAR-100 [77] contains a `dolphin` class. However, images in CIFAR-100 are only 32x32 pixels in size, too small to be useful for the task at hand.

Moving the search away from generalised datasets and towards those which are targetted at conservation efforts or the natural environment also proved fruitless. A large portion of these datasets focus on camera traps or land-based fauna, such as iWildCam [8], for reasons discussed in more detail in Section 2.4. Some images included in the iNaturalist dataset [160] are of cetaceans, such as a class for the short-beaked common dolphin (*Delphinus delphis*), however most focus on other aquatic animals such as the Florida manatee (*Trichechus manatus*), various amphibians, and molluscs.

3.3.1 The Zanzibar Dataset

Due to the lack of open-source and published datasets to aid in the development of this cetacean detector, one was required to be created. As the focus of this project as a whole was the utilisation of the developed system to aid in conservation efforts of resident cetacean populations off the Northumberland coastline, ideally the created dataset would come from this area. At the time of initial testing however this was not possible due to a lack of available data from the survey area.

As such, alternative data was provided by the Marine MEGAfauna Lab, obtained during a 2015 conservation effort undertaken in Zanzibar, Tanzania. The catalogue provided consisted of 1021 images of size 5184x3456, and was supplied in a format suitable for manual photo-identification rather than for the training of a neural network. Work was then undertaken to convert this conservation catalogue into a machine learning dataset.

In order to convert this catalogue into a machine learning dataset, the provided images must first be labelled. This was achieved using the VGG Image Annotator software, known as VIA [41]. Other labelling software such as LabelImg [155] were examined, however VIA was deemed the best choice for the task at hand. This software was chosen for multiple reasons; first, the software is noticeably easy to use and allows for efficient labelling on a per-pixel basis as required by Mask-RCNN. Second, the tutorial data provided by the Mask-RCNN Github repository was labelled in VIA format, showing that this code implementation

would accept data labelled in this format. Furthermore, use cases of VIA being utilised for labelling of marine-oriented data are available in literature [110], providing evidence of suitability of the labelling software for research purposes and data representing similar conditions.

Before labelling the Tanzania data, some curation was performed. Each image labelled by VIA is required to contain at least one non-background class. As such, any images provided which did not contain an example of a `dolphin` class were discarded. Other images where the class examples were unsuitable for training a Mask-RCNN model, such as those which contained only an extremely small section of the photographed dolphin or were deemed too blurry, were also removed. This left 312 images which were suitable for the Mask-RCNN.

The process for labelling the data with VIA is rather straightforward. The software runs locally through a web browser, with each image labelled sequentially. Figure 3.6 shows an example image labelled using VIA and the resulting JSON created for the image. Each image is shown on-screen to the user who is then able to trace around class examples by selecting multiple points on the image. Once a full trace has been performed, any pixels inside of the trace are treated as one class. This class is labelled through the use of a class attribute, in the case of the Tanzania data this was the class label `fin`, denoting the class example as a fin above the waterline. These labels are stored in a corresponding JSON file, which is fed to the Mask-CNN model along with the images during training. This labelling allows the model to learn per-pixel class examples during training. This tracing method also allows for each distinct individual in a group to be labelled individually, even if overlapping, which would be much harder to perform with bounding box labelling and allows the model to learn how to differentiate between group members.

Once all 312 images had been labelled, it was then possible to create a train-test split. The 312 images were divided using an 80-20 split, where 80% of the images are designated for training the Mask-RCNN model, known as the training set, and the remaining 20% were held back for model evaluation, known as the test set. By evaluating on previously unseen data this affords researchers the ability to understand the generalisability of the trained model, mitigating overfitting.

This newly created Zanzibar dataset would allow for prototyping to begin, determining the suitability of a Mask-RCNN based model for the task of a cetacean detector. The Zanzibar dataset was very similar in content to what would be expected from a dataset created from North Sea survey data once this had taken place and thus gave a good baseline for experimentation.

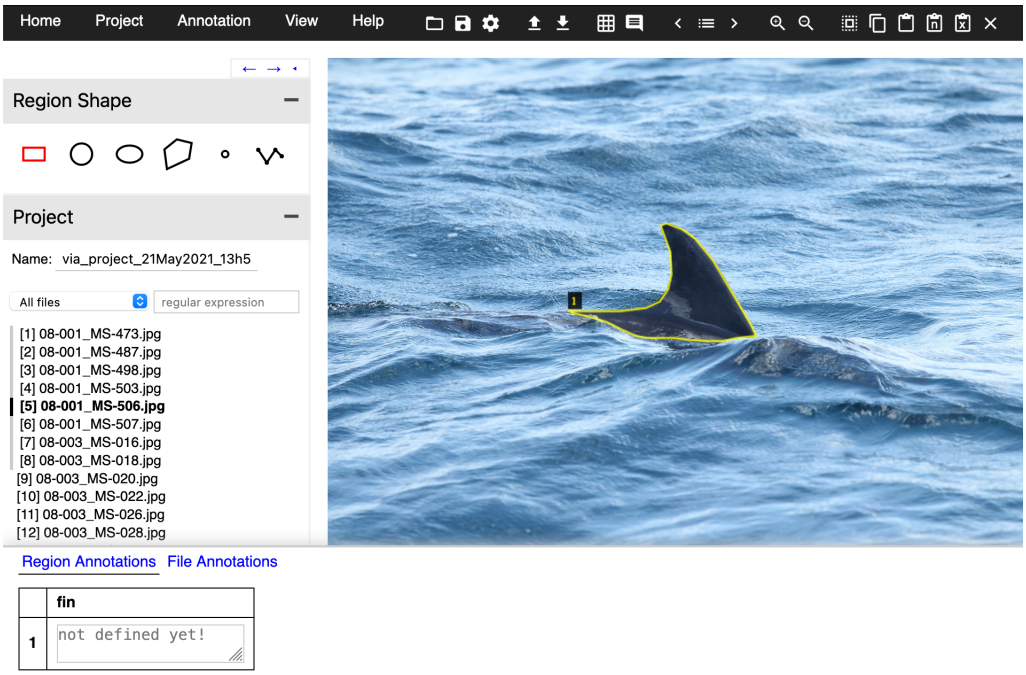

Figure 3.6 An example image showing the labelling processes using VIA.

3.3.2 Transfer Learning

Whilst the Zanzibar dataset provides experimental data similar to that which the Mask-RCNN model will be required to process, the amount of data is extremely small. Deep learning models often require thousands of images when training to produce generalisable and accurate models. As such, this dataset alone would not be enough to train the cetacean detector. One way to fix this issue would be to locate more photo-id data. However, little extra data was readily available from the Marine MEGAfauna Lab, and data from other labs would require a large amount of effort to obtain. Cetacean catalogues are closely guarded by conversation labs due to the large amount of effort required to obtain them. Second, any further data collected would also need to be labelled and incorporated into the now existing dataset, which again would require significant time and effort. These issues rendered the prospect of expanding the Zanzibar dataset unachievable in the time required.

Another available fix for this issue is the concept of transfer learning. This is a technique whereby models trained to perform one task are repurposed to aid in a second, usually more specialised task. These initial models have typically been trained on large generalised datasets such as ImageNet [37] or Microsoft's Common Objects in Context, more commonly

known as MSCOCO [89]. These datasets often contain hundreds of thousands of images covering a large number of classes, which make them perfect for the task of transfer learning.

By first training a model on these large datasets, the model is able to learn the basics of image understanding, for example the concept of basic shapes and colour, allowing for the development of a generic visual understanding model. By utilising these models, we effectively provide our own model with a head-start in its learning process, there is no need to utilise the small amount of data available in the Zanzibar dataset for low level learning; it can instead be saved for allowing the model to understand and generalise to the domain specific task, such as cetacean detection. For a more in-depth analysis of transfer learning, see Pan *et al.* [115].

Training a neural network, or model, is extremely computationally and time expensive due to the large dataset sizes used. As such, many models suitable for transfer learning can be obtained in a pre-trained state. These pre-trained models are hosted by model zoos, which provide frozen model weight files in a format which allow for transfer learning to take place through a process known as fine-tuning. Here, a model from the zoo is downloaded and n-number of deeper layers are unfrozen. Next, additional layers are added to the model which perform the domain specific task. The unfrozen layers and the additional layers are then trained on the domain specific task, allowing for the fine-tuning of the higher-level feature extraction.

3.3.3 Utilising Transfer Learning to Train the Mask-RCNN

The use of transfer learning can be easily adapted for the training of the cetacean detector for use with the Zanzibar dataset, achieved directly through Tensorflow. First, a backbone model is chosen. For the cetacean detector, it was decided that a ResNet50 backbone would be utilised. Matterport's Mask-RCNN implementation allows for the use of a ResNet50 or ResNet101 backbone, both standard variants of ResNet which are 50 and 101 layers deep respectively. ResNet50 was chosen over ResNet101 as during initial experimentation with the Matterport provided tutorial data, no significant improvement in accuracy was achieved using the deeper 101 layer model however this greatly increased training time as double the number of layers were required to be trained.

Once a backbone has been chosen, it is then loaded into Tensorflow. Next, the pre-trained model weights are downloaded from the model zoo. These weights denote the strength of the connections between the model's layers, and when Tensorflow initially loads in the backbone model the weights of each layer are randomly initialised. During a normal training run, where the backbone model is trained from scratch, these randomly initialised weights would be manipulated through the use of backpropagation; the weights of each neuron in the layer

changed to reflect how much effect said neuron has on the output of the layer in order to achieve the desired overall model output (such as a pixel classification).

In transfer learning however, these initial weights are overwritten by those downloaded from the model zoo. This replicates the state of the model trained on the larger dataset onto the initial backbone model; all of the weights are frozen into their final pre-trained state. As previously mentioned, there are multiple different models available in the zoo usually trained on large benchmark datasets. As such, before utilising transfer learning it is important to make an informed decision about which benchmark dataset the pre-trained model has itself been trained on. For this project it was decided that the ResNet50 weights trained on MSCOCO would be utilised. This was due to the fact that MSCOCO is primarily an instance segmentation dataset, and thus one of the most appropriate to use for transfer learning to another instance segmentation task. The use of MSCOCO for pre-training on Mask-RCNN has in recent years been well documented in literature for a variety of tasks [35, 48, 180]. When utilising an MSCOCO pre-trained backbone for a Mask-RCNN based task, it is important to note that certain layers must be excluded when loading in the pre-trained weights as these are only utilised in Mask-RCNN models, such as those which deal with the per-pixel masks. This is because these layers require a matching number of classes; if the MSCOCO weights were utilised here there would be a mismatch between the number of classes in MSCOCO (80) and in the Zanzibar dataset (1).

Once the backbone model has been loaded with pre-trained weights, the total number of layers to fine-tune must be decided. This can be considered similar to a hyperparameter, as it must be chosen at run time by the user. Whilst any number of the layers can be chosen for fine-tuning, generally either all of the model layers or only the heads; these are layers required for the Region Proposal Network, the pixel classification, and masking layers of the model. For the purposes of hyperparameter tuning, the number of unfrozen layers can be included in a grid search.

3.3.4 Data Augmentation

As well as transfer learning, the use of data augmentation was also explored to help mitigate the issue of dataset size. This technique allows for datasets to be artificially expanded by performing random perturbations to each data point which are then automatically class labelled the same as the original input.

When augmenting data, it is extremely important to understand a dataset's problem domain to ensure that the any transformations are realistic and expose the training model to data which, whilst not present in the dataset before augmentation, could still reasonably expected to be seen by the model when deployed. Further, augmentation must only occur on

the training data, and not the test data. This is in contrast to preprocessing techniques such as resizing, which must occur to all data points.

As the Zanzibar data contains relatively few images, it is a prime candidate for data augmentation. This can be performed in one of two ways; in either an offline or an online manner. In offline data augmentation, the entire train split is augmented before the images are passed to the model, occurring as a preprocessing step. This is extremely useful for very small datasets where the number of examples needs to be increased before the model can begin training, or if training time is a concern. The major issue with offline augmentations however is that, because the data is perturbed and then passed to the model, offline augmentation produces a fixed number of augmented images. This can be solved with online augmentation, which occurs in real-time as the model trains. The model is passed the original, unperturbed data which is then augmented during each training batch. This allows for the model to see a potentially unlimited number of new images, as each input image is randomly perturbed before being used for training. Once training on the batch has been completed, the augmented images are discarded and new perturbations performed on the original images. As such, online augmentation is, if possible, greatly preferred and allows for a much higher chance of model generalisation.

Whilst the Zanzibar dataset is small compared to others used for deep learning, it is large enough to allow for online augmentation. In order to begin testing the effect of data augmentation on the Mask-RCNN training process, two different augmentation strategies were created which contained unique workflows.

The first strategy, *aug1*, selected between zero and three of the following perturbations: (1) *horizontal flip*: flip the image horizontally with a probability of 0.5, (2) *vertical flip*: flip the image vertically with a probability of 0.5, (3) *rotation*: rotate the image either 90, 180, or 270 degrees each with equal probability of occurring, (4) *scaling*: scale the image between 80% and 120% on both axis independent of each other, (5) *brightness*: multiply all pixels in the image with a random value between 0.8 and 1.5, (6) *Gaussian blur*: blur the image with a Gaussian kernel with radius randomly assigned between 0 and 5.

The second strategy, *aug2*, was more complex, performing the following perturbations in a sequentially random order on 67% of the images only: (1) *horizontal flip*: flip the image horizontally with a probability of 0.5, (2) *cropping*: crop each side of the image randomly between 0% and 10% of the total side length, (3) *Gaussian blur*: blur the image with a Gaussian kernel with radius randomly assigned between 0 and 2.5, with a probability of blurring of 0.5, (4) *contrast*: strengthen or weaken the contrast of the image by a random factor between 0.75 and 1.5, (5) *additive Gaussian noise*: sample the noise per channel - adding noise to the colour of the pixels, (6) *brightness*: multiply all pixels in the image with

a random value between 0.8 and 1.2, (7) *scaling*: scale the image between 80% and 120% on both axis independent of each other, (8) *rotation*: rotate the image randomly between -180 and 180 degrees.

The use of two augmenters allowed for evaluation on whether a simple or more complex augmentation strategy would be appropriate for this use case. By using multiple augmenters we can treat them as a hyperparameter of model training, allowing the augmenter chosen to be added to the search space.

3.3.5 Mask-RCNN Hyperparameter Tuning

When training a Mask-RCNN model there are a large range of hyperparameters, or user defined values, which must be set before training can occur. These can be broken down into two subgroups; detection hyperparameters influence the output of the model, and training hyperparameters which influence the training of the model. Thankfully most deep learning frameworks provide default values for most, if not all hyperparameters. These default values are known to work well regardless of dataset or task, and so many have been used when training the Mask-RCNN. Some hyperparameters however can have a large effect on the final model, and so an exploration of the optimal value for these has been undertaken.

With regards to the detection hyperparameters, only the minimum confidence of the model was changed from the default of 0.7 to 0.9. This was changed as during initial trials it was found models trained on the Zanzibar data would often produce a high number of false positives (for example detecting a wave as a fin) or create duplicate detections (one fin detected twice). By increasing the minimum confidence of the model to 0.9, we increase the threshold at which the model returns a detection to 90%, or in other words for every detection the model must be 90% sure that the detection is actually a fin. This reduced both the false positive rate and duplicate detection rate of the model.

The vast majority of hyperparameters are those which influence the training process. Selection of the optimal hyperparameters is an extremely computationally and time expensive task, as the optimal values of the hyperparameters are not known before training begins. Indeed, even after training has finished and a model which produces satisfactory results has been found there is no guarantee that the hyperparameters of this model are the best, just that they were the best which have been found so far. As such, in order to determine the best hyperparameters for a given model and task, the search space of all possible hyperparameters must be searched. This is often infeasible due to time and resource constraints however, and as such techniques such as a grid search will be performed. During a grid search, each of the possible hyperparameters will have a range of values defined. A model is then trained using the data and every possible combination of hyperparameters. Once each model has been

trained, they are then evaluated to determine the best model. In some cases an acceptable model will be found during the initial grid search, however this may not be the case. In this situation however the previous grid search may not be useless, as it may provide insight into how to refine the search space to increase the chances of an acceptable model being trained. This process of finding the optimal hyperparameters for the model is known as hyperparameter tuning.

Learning Rate Scheduling & Optimisers

One of the most important hyperparameters to tune is the learning rate, which dictates how much the weights of the model should change in response to the estimated error calculated during backpropagation. If the learning rate is too large this will lead to an unstable training process whereby gradient descent can never reach the minimum value but rather bounce either side of it. If the learning rate is set too small however the training process will take an extremely long time to converge. In order to help the model reach its optimal minima in a reasonable time, the learning rate can be scheduled using a scheduler. These allow for the learning rate the be decreased when some criteria is met, such as after a set number of epochs, allowing for larger weight changes initially for fast training before reducing the descent steps as time goes on, decreasing the chance of gradient descent jumping over the minima.

As well as learning rate schedulers, adaptive learning rate optimisers can also be used. These optimisers provide an alternative to vanilla SGD and are capable of adapting to the dataset it is given and the current training process, changing the learning rate without a defined schedule. This often allows for a more optimised and efficient training process when compared to using vanilla SGD, as discussed in Section 2.2.2. During hyperparameter tuning of the Mask-RCNN, two optimisers were chosen for evaluation.

The first, SGD with restarts (SGDR) [96] allows for the decrease in the learning rate through a process known as cosine annealing, whereby the decrease follows a cosine waveform. This allows for a high starting learning rate allowing for a fast approach to a local minima before reducing the rate as the number of epochs increases to prevent a jump over the minima, similarly to how a scheduler works. However it may not be the case that this local minima is the global minima, the lowest possible point in the space. Due to cosine annealing it would not be possible to leave the local minima, the learning rate needs to be increased again to allow for this. As such the learning rate is *restarted*, or increased back to its maximum, to allow for the training process to jump out of the local minima; if it is indeed the case that this local minima is also the global minima then the training process will return to the point it was at before the restart, however if the local minima was not the

global minima, the restart will allow for the training process to leave the sub-optimal minima it previously found.

The second learning rate optimisation explored during hyperparameter tuning is Adam, or adaptive moment estimation. This optimiser is extremely popular in the world of deep learning thanks to its ability to achieve impressive results in relatively short training times. This is possible through the use of one learning rate for each model weight, in contrast to the singular learning rate for the whole model as seen in SGD or SGDR. Adam also utilises parts of other optimisers such as AdaGrad [39] and RMSProp [153] to allow the optimiser to work well with both sparse and noisy data. For a complete breakdown of the inner workings of Adam, see Kingma *et al.* [76].

Weight Decay

The goal of neural network development is to utilise the training data in such a way that the resulting generated model performs well on unseen data. In order for this to be achieved our model must be generalisable, having learnt enough from the training data to perform well at the given task but not having learnt so well that it is unable to perform adequately on unseen data. If a model fails to generalise, it is said to have overfitted the data. For example lets say we wish to develop a cat detector, a model which given an input image will tell you if there is a cat present. However, we only train our model on images with white cats in them. The model trains well, and is always able to tell you if there is a cat in the images it sees during training. When deployed, the model fails to identify any images containing black cats - the model has learnt the training data too well and believes cats can only be white; the model has overfitted.

There are many different techniques to reduce overfitting in neural networks, one of the easiest is to simply collect more training data. However as previously mentioned, due to how closely guarded cetacean photo-id catalogue data is, this was not possible. As such, the use of weight decay was explored during hyperparameter tuning. Weight decay is a regularisation technique which allows the model training to be penalised in proportion to the size of its weights. This incentivises the training process to keep the weights small, which has been shown to improve generalisation to unseen data [79]. As the Zanzibar dataset is comparability small compared to the usual size of datasets for this task, allowing the model to generalise well using small amounts of data is extremely important.

RPN Anchor Scales

As discussed in Section 2.3.2, Region Proposal Networks (RPNs) can be utilised in object detection due to their ability to determine potential regions of interest (RoIs) in the image, known as anchors. These anchors are then classified as either background or of a learnable class, such as `dolphin`. To allow the RPN to be object-size invariant, anchor scales are utilised. These scales, provided as a list of values which correspond to the square anchor side in pixels, determine what sizes the RoIs proposed by the RPN should be. For example, let's say an anchor scale of `[32]` is passed to the RPN, this would mean that all RoIs proposed by the RPN would be of size 32x32 pixels. As such, the anchor scale provided to the RPN can be considered a hyperparameter, as it must be determined the best scales for the RPN to propose to allow for the detection of object regardless of their size.

Hyperparameter Tuning via a Grid Search

Although only a few hyperparameters have been chosen to tuned, the size of the possible search space to evaluate is still extremely large. As mentioned previously, it is not feasible both from a time and resource perspective to evaluate the entire space and find the truly optimal value for each hyperparameter. Instead the search space is discretised using a grid search, for each hyperparameter a subset of possible values is selected. Each combination of possible hyperparameter values is then evaluated to determine which set of values produces a satisfactory model. The list of possible hyperparameter combinations and model name, determined by the datetime value at the start of the model's first training run, can be seen in Table

- Grid search, what values were used (table?)
- Evaluation of the models (metrics used, graphs, deciding on best model)
- Testing the Mask-RCNN on NDD when trained on Zanzibar data - Training a Mask-RCNN on NDD (?)

Appendix

Mask-RCNN Grid Search Hyperparameter Combinations

Model Name	Weight Decay	RPN Anchor Scales	Optimiser	Augmentation Strategy	Pre-train
20190829T1458	0.01	(16, 32, 64, 128, 256)	Adam	aug1	False
20190829T2020	0.01	(16, 32, 64, 128, 256)	Adam	aug2	False
20190830T0145	0.01	(16, 32, 64, 128, 256)	Adam	None	False
20190830T0714	0.01	(16, 32, 64, 128, 256)	Adam	aug1	True
20190830T1443	0.01	(16, 32, 64, 128, 256)	Adam	aug2	True
20190830T2019	0.01	(16, 32, 64, 128, 256)	Adam	None	True
20190902T0946	0.01	(16, 32, 64, 128, 256)	SGDR	aug1	False
20190904T2004	0.01	(16, 32, 64, 128, 256)	SGDR	None	False
20190905T1813	0.001	(32, 64, 128, 256, 512)	SGDR	aug1	False
20190905T1826	0.01	(16, 32, 64, 128, 256)	SGDR	aug2	False
20190905T2202	0.001	(32, 64, 128, 256, 512)	Adam	None	False
20190905T2336	0.001	(32, 64, 128, 256, 512)	Adam	aug1	False
20190906T0332	0.001	(32, 64, 128, 256, 512)	SGDR	None	False
20190906T0851	0.01	(32, 64, 128, 256, 512)	Adam	None	False
20190907T0932	0.001	(16, 32, 64, 128, 256)	Adam	aug1	False
20190907T0933	0.0001	(32, 64, 128, 256, 512)	Adam	aug2	True
20190907T0934	0.01	(32, 64, 128, 256, 512)	SGDR	None	False
20190907T1451	0.001	(16, 32, 64, 128, 256)	Adam	None	False
20190907T1500	0.0001	(32, 64, 128, 256, 512)	Adam	aug1	True
20190907T1545	0.01	(32, 64, 128, 256, 512)	Adam	aug2	True
20190907T2026	0.0001	(32, 64, 128, 256, 512)	SGDR	None	False
20190907T2126	0.001	(16, 32, 64, 128, 256)	Adam	aug1	True
20190907T2215	0.001	(16, 32, 64, 128, 256)	SGDR	aug2	False
20190908T0202	0.0001	(16, 32, 64, 128, 256)	Adam	None	False
20190908T0352	0.01	(32, 64, 128, 256, 512)	Adam	aug2	False
20190908T0417	0.0001	(32, 64, 128, 256, 512)	Adam	aug1	False
20190908T0957	0.0001	(32, 64, 128, 256, 512)	Adam	None	True
20190908T1102	0.0001	(32, 64, 128, 256, 512)	Adam	aug2	False
20190908T1204	0.001	(16, 32, 64, 128, 256)	SGDR	aug1	False
20190908T1939	0.001	(16, 32, 64, 128, 256)	Adam	aug2	True
20190908T2043	0.0001	(32, 64, 128, 256, 512)	SGDR	aug1	False
20190908T2139	0.0001	(16, 32, 64, 128, 256)	Adam	None	True
20190909T0723	0.0001	(16, 32, 64, 128, 256)	Adam	aug1	True
20190911T1922	0.01	(16, 32, 64, 128, 256)	Adam	aug2	True
20190912T0045	0.0001	(16, 32, 64, 128, 256)	Adam	aug2	True
20190912T0608	0.0001	(16, 32, 64, 128, 256)	SGDR	aug1	False
20191101T1633	0.0001	(32, 64, 128, 256, 512)	SGDR	aug2	False
20191101T2104	0.001	(8, 16, 32, 64, 128)	SGDR	aug2	False
20191102T0140	0.01	(32, 64, 128, 256, 512)	SGDR	aug1	False
20191102T0615	0.01	(8, 16, 32, 64, 128)	SGDR	aug2	False
20191102T1051	0.0001	(16, 32, 64, 128, 256)	SGDR	None	False
20191102T1528	0.001	(32, 64, 128, 256, 512)	SGDR	aug2	False
20191102T2006	0.0001	(8, 16, 32, 64, 128)	SGDR	aug2	False
20191103T0044	0.0001	(8, 16, 32, 64, 128)	SGDR	aug1	False
20191103T0520	0.001	(8, 16, 32, 64, 128)	SGDR	None	False
20191103T0959	0.01	(32, 64, 128, 256, 512)	SGDR	aug2	False
20191103T1441	0.0001	(8, 16, 32, 64, 128)	SGDR	None	False
20191103T1921	0.0001	(16, 32, 64, 128, 256)	SGDR	aug2	False
20191104T0011	0.001	(8, 16, 32, 64, 128)	SGDR	aug1	False

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., and Isard, M. (2016). Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

[2] Alber, M., Bello, I., Zoph, B., Kindermans, P.-J., Ramachandran, P., and Le, Q. (2018). Backprop Evolution. *arXiv:1808.02822 [cs, stat]*. arXiv: 1808.02822.

[3] Anantharaman, R., Velazquez, M., and Lee, Y. (2018). Utilizing Mask R-CNN for Detection and Segmentation of Oral Diseases. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2197–2204.

[4] Arnbom, T. (1987). Individual identification of sperm whales. *Report of the International Whaling Commission*, 37(20):201–204.

[5] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv:1511.00561 [cs]*. arXiv: 1511.00561.

[6] Baird, R. W., Gorgone, A. M., McSweeney, D. J., Ligon, A. D., Deakos, M. H., Webster, D. L., Schorr, G. S., Martien, K. K., Salden, D. R., and Mahaffy, S. D. (2009). Population structure of island-associated dolphins: Evidence from photo-identification of common bottlenose dolphins (Tursiops truncatus) in the main Hawaiian Islands. *Marine Mammal Science*, 25(2):251–274. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1748-7692.2008.00257.x.

[7] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359.

[8] Beery, S., Morris, D., and Perona, P. (2019a). The iWildCam 2019 Challenge Dataset. *arXiv:1907.07617 [cs]*. arXiv: 1907.07617.

[9] Beery, S., Morris, D., and Yang, S. (2019b). Efficient Pipeline for Camera Trap Image Review. *arXiv:1907.06772 [cs]*. arXiv: 1907.06772.

[10] Bengio, S., Bengio, Y., and Cloutier, J. (1994). Use of genetic programming for the search of a new learning rule for neural networks. pages 324–327. IEEE.

[11] Berger-Wolf, T. Y., Rubenstein, D. I., Stewart, C. V., Holmberg, J. A., Parham, J., Menon, S., Crall, J., Van Oast, J., Kiciman, E., and Joppa, L. (2017). Wildbook: Crowdsourcing, computer vision, and data science for conservation. *arXiv:1710.08880 [cs]*. arXiv: 1710.08880.

[12] Bevan, E., Whiting, S., Tucker, T., Guinea, M., Raith, A., and Douglas, R. (2018). Measuring behavioral responses of sea turtles, saltwater crocodiles, and crested terns to drone disturbance to define ethical operating thresholds. *PLOS ONE*, 13(3):e0194460. Publisher: Public Library of Science.

[13] Bigg, M. (1982). An Assessment of Killer Whale (Orcinus orca) Stocks off Vancouver Island, British Columbia. *Report of the International Whaling Commission*, 32(65):12.

[14] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934 [cs, eess]*. arXiv: 2004.10934.

[15] Bogucki, R., Cygan, M., Khan, C. B., Klimek, M., Milczek, J. K., and Mucha, M. (2019). Applying deep learning to right whale photo identification. *Conservation Biology*, 33(3):676–684. _eprint: https://conbio.onlinelibrary.wiley.com/doi/pdf/10.1111/cobi.13226.

[16] Bolya, D., Zhou, C., Xiao, F., and Jae Lee, Y. (2019). [1904.02689] YOLACT: Real-time Instance Segmentation.

[17] Bottou, L. and Bousquet, O. (2008). The Tradeoffs of Large Scale Learning. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 161–168. Curran Associates, Inc.

[18] Bouma, S., Pawley, M. D., Hupman, K., and Gilman, A. (2018a). Individual Common Dolphin Identification Via Metric Embedding Learning. In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. ISSN: 2151-2191.

[19] Bouma, S., Pawley, M. D. M., Hupman, K., and Gilman, A. (2018b). Individual common dolphin identification via metric embedding learning. page 7.

[20] Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A Theoretical Analysis of Feature Pooling in Visual Recognition. page 8.

[21] Buhrmester, M., Kwang, T., and Gosling, S. D. (2011). Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5.

[22] Buric, M., Pobar, M., and Ivasic-Kos, M. (2018). Ball Detection Using Yolo and Mask R-CNN. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 319–323.

[23] Burke, C. J., Aleo, P. D., Chen, Y.-C., Liu, X., Peterson, J. R., Sembroski, G. H., and Lin, J. Y.-Y. (2019). Deblending and classifying astronomical sources with Mask R-CNN deep learning. *Monthly Notices of the Royal Astronomical Society*, 490(3):3952–3965. Publisher: Oxford Academic.

[24] Caldwell, D. K. (1955). Evidence of Home Range of an Atlantic Bottlenose Dolphin. *Journal of Mammalogy*, 36(2):304–305.

[25] Centelleghe, C., Carraro, L., Gonzalvo, J., Rosso, M., Esposti, E., Gili, C., Bonato, M., Pedrotti, D., Cardazzo, B., Povinelli, M., and Mazzariol, S. (2020). The use of Unmanned Aerial Vehicles (UAVs) to sample the blow microbiome of small cetaceans. *PLOS ONE*, 15(7):e0235537. Publisher: Public Library of Science.

[26] Cheeseman, T. (2019). Happywhale Presentation - World Marine Mammal Conference 2019 - Rise of the Machines Workshop (available at: drive.google.com/file/d/1yRtNKagANZOgpY6852zzUpZmiFXuZmLM).

[27] Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., and Yan, Y. (2020). BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation. *arXiv:2001.00309 [cs]*. arXiv: 2001.00309.

[28] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2014). Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv:1412.7062 [cs]*. arXiv: 1412.7062.

[29] Cheney, B., Thompson, P. M., Ingram, S. N., Hammond, P. S., Stevick, P. T., Durban, J. W., Culloch, R. M., Elwen, S. H., Mandleberg, L., Janik, V. M., Quick, N. J., ISLAS-Villanueva, V., Robinson, K. P., Costa, M., Eisfeld, S. M., Walters, A., Phillips, C., Weir, C. R., Evans, P. G. H., Anderwald, P., Reid, R. J., Reid, J. B., and Wilson, B. (2013). Integrating multiple data sources to assess the distribution and abundance of bottlenose dolphins Tursiops truncatus in Scottish waters. *Mammal Review*, 43(1):71–88. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2907.2011.00208.x.

[30] Chiao, J.-Y., Chen, K.-Y., Liao, K. Y.-K., Hsieh, P.-H., Zhang, G., and Huang, T.-C. (2019). Detection and classification the breast tumors using mask R-CNN on sonograms. *Medicine*, 98(19).

[31] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015). The Loss Surfaces of Multilayer Networks. page 13.

[32] Chu, P., Li, Z., Lammers, K., Lu, R., and Liu, X. (2020). DeepApple: Deep Learning-based Apple Detection using a Suppression Mask R-CNN. *arXiv:2010.09870 [cs]*. arXiv: 2010.09870.

[33] Connor, R. C. and Krützen, M. (2015). Male dolphin alliances in Shark Bay: changing perspectives in a 30-year study. *Animal Behaviour*, 103:223–235.

[34] Constantine, R., Jackson, J. A., Steel, D., Baker, C. S., Brooks, L., Burns, D., Clapham, P., Hauser, N., Madon, B., Mattila, D., Oremus, M., Poole, M., Robbins, J., Thompson, K., and Garrigue, C. (2012). Abundance of humpback whales in Oceania using photo-identification and microsatellite genotyping. *Marine Ecology Progress Series*, 453:249–261.

[35] Couteaux, V., Si-Mohamed, S., Nempont, O., Lefevre, T., Popoff, A., Pizaine, G., Villain, N., Bloch, I., Cotten, A., and Boussel, L. (2019). Automatic knee meniscus tear detection and orientation classification with Mask-RCNN - ScienceDirect. *Diagnostic and Interventional Imaging*, 100(4):235–242.

[36] Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2933–2941. Curran Associates, Inc.

[37] Deng, J., Dong, W., Socher, R., Li, L.-J., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, Miami, FL. IEEE.

[38] Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *arXiv:1801.07698 [cs]*. arXiv: 1801.07698.

[39] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. page 39.

[40] Dumoulin, V. and Visin, F. (2018). [1603.07285] A guide to convolution arithmetic for deep learning.

[41] Dutta, A. and Zisserman, A. (2019). The VIA Annotation Software for Images, Audio and Video. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2276–2279, Nice France. ACM.

[42] Elsken, T., Metzen, J. H., and Hutter, F. (2018). Neural Architecture Search: A Survey. *arXiv:1808.5377 null*. arXiv: 1808.5377.

[43] Feyrer, L. J., Stewart, M., Yeung, J., Soulier, C., and Whitehead, H. (2021). Origin and Persistence of Markings in a Long-Term Photo-Identification Dataset Reveal the Threat of Entanglement for Endangered Northern Bottlenose Whales (Hyperoodon ampullatus). *Frontiers in Marine Science*, 8. Publisher: Frontiers.

[44] Fiori, L., Martinez, E., Orams, M. B., and Bollard, B. (2020). Using Unmanned Aerial Vehicles (UAVs) to assess humpback whale behavioral responses to swim-with interactions in Vava'u, Kingdom of Tonga. *Journal of Sustainable Tourism*, 28(11):1743–1761. Publisher: Routledge _eprint: https://doi.org/10.1080/09669582.2020.1758706.

[45] Fisheries, N. (2018). FinBase Photo-Identification Database System | NOAA Fisheries.

[46] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. (2020). Sharpness-Aware Minimization for Efficiently Improving Generalization. *arXiv:2010.01412 [cs, stat]*. arXiv: 2010.01412.

[47] Forney, K. A. and Barlow, J. (1998). Seasonal Patterns in the Abundance and Distribution of California Cetaceans, 1991–1992. *Marine Mammal Science*, 14(3):460–489. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1748-7692.1998.tb00737.x.

[48] Fujita, H., Itagaki, M., Ichikawa, K., Hooi, Y. K., Kawahara, K., and Sarlan, A. (2020). Fine-tuned Surface Object Detection Applying Pre-trained Mask R-CNN Models. In *2020 International Conference on Computational Intelligence (ICCI)*, pages 17–22.

[49] Galatius, A. and Kinze, C. C. (2016). Lagenorhynchus albirostris (Cetacea: Delphinidae). *Mammalian Species*, 48(933):35–47.

[50] Gavves, E., Fernando, B., Snoek, C. G. M., Smeulders, A. W. M., and Tuytelaars, T. (2013). Fine-Grained Categorization by Alignments. pages 1713–1720.

[51] Gibson, C. E., Williams, D., Dunlop, R., and Beck, S. (2020). Using social media as a cost-effective resource in the photo-identification of a coastal bottlenose dolphin community. *Aquatic Conservation: Marine and Freshwater Ecosystems*, 30(8):1702–1710. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aqc.3356.

[52] Giles, A. B., Butcher, P. A., Colefax, A. P., Pagendam, D. E., Mayjor, M., and Kelaher, B. P. (2020). Responses of bottlenose dolphins (Tursiops spp.) to small drones. *Aquatic Conservation: Marine and Freshwater Ecosystems*, n/a(n/a). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aqc.3440.

[53] Gilman, A., Hupman, K., Stockin, K., and Pawley, M. D. M. (2016). Computer-assisted recognition of dolphin individuals using dorsal fin pigmentations - IEEE Conference Publication.

[54] Girshick, R. (2015). Fast R-CNN. *arXiv:1504.08083 [cs]*. arXiv: 1504.08083.

[55] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. pages 580–587.

[56] Gray, P. C., Bierlich, K. C., Mantell, S. A., Friedlaender, A. S., Goldbogen, J. A., and Johnston, D. W. (2019). Drones and convolutional neural networks facilitate automated and accurate cetacean species identification and photogrammetry. *Methods in Ecology and Evolution*, 10(9):1490–1500. _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13246.

[57] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 Object Category Dataset.

[58] Hale, S. A. (2012). Unsupervised Threshold for Automatic Extraction of Dolphin Dorsal Fin Outlines from Digital Photographs in DARWIN (Digital Analysis and Recognition of Whale Images on a Network). *arXiv:1202.4107 [cs]*. arXiv: 1202.4107.

[59] Hammond, P. S., Macleod, K., Berggren, P., Borchers, D. L., Burt, L., Cañadas, A., Desportes, G., Donovan, G. P., Gilles, A., Gillespie, D., Gordon, J., Hiby, L., Kuklik, I., Leaper, R., Lehnert, K., Leopold, M., Lovell, P., Øien, N., Paxton, C. G. M., Ridoux, V., Rogan, E., Samarra, F., Scheidat, M., Sequeira, M., Siebert, U., Skov, H., Swift, R., Tasker, M. L., Teilmann, J., Canneyt, O. V., and Vázquez, J. A. (2013). Cetacean abundance and distribution in European Atlantic shelf waters to inform conservation and management. *Biological Conservation*, 164:107 – 122.

[60] Hammond, P. S., Mizroch, S. A., and Donovan, G. P. (1990). Individual recognition of cetaceans: use of photo-identification and other techniques to estimate population parameters: incorporating the proceedings of the symposium and workshop on individual recognition and the estimation of cetacean population parameters. *International Whaling Commission*, 12.

[61] Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. (2014). Simultaneous Detection and Segmentation. *arXiv:1407.1808 [cs]*. arXiv: 1407.1808.

[62] Harislqbal88 (2018). PlotNeuralNet.

[63] Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

[64] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. *arXiv:1703.06870 [cs]*. arXiv: 1703.06870.

[65] He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*. arXiv: 1512.03385.

[66] He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*. arXiv: 1502.01852.

[67] Hecht-nielsen, R. (1992). III.3 - Theory of the Backpropagation Neural Network**Based on "nonindent" by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE. In Wechsler, H., editor, *Neural Networks for Perception*, pages 65–93. Academic Press.

[68] Hillman, G., Kehtarnavaz, N., Wursig, B., Araabi, B., Gailey, G., Weller, D., Mandava, S., and Tagare, H. (2002). "Finscan", a computer system for photographic identification of marine animals. In *Proceedings of the Second Joint 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society] [Engineering in Medicine and Biology*, volume 2, pages 1065–1066 vol.2. ISSN: 1094-687X.

[69] Holmberg, J., Norman, B., and Arzoumanian, Z. (2009). Estimating population size, structure, and residency time for whale sharks Rhincodon typus through collaborative photo-identification. *Endangered Species Research*, 7(1):39–53.

[70] Hong, J., Fulton, M., and Sattar, J. (2020). TrashCan: A Semantically-Segmented Dataset towards Visual Detection of Marine Debris. *arXiv:2007.08097 [cs]*. arXiv: 2007.08097.

[71] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K. (2017). Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297, Honolulu, HI. IEEE.

[72] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*. arXiv: 1502.03167.

[73] Kaggle (2018). Humpback Whale Identification Challenge.

[74] Karnowski, J., Hutchins, E., and Johnson, C. (2015). Dolphin Detection and Tracking. In *2015 IEEE Winter Applications and Computer Vision Workshops*, pages 51–56, Waikoloa, HI, USA. IEEE.

[75] Khosla, A., Jayadevaprakash, N., Yao, B., and Li, F.-F. (2011). Novel Dataset for Fine-Grained Image Categorization: Stanford Dogs. page 2.

[76] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. arXiv: 1412.6980.

[77] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. page 60.

[78] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[79] Krogh, A. and Hertz, J. A. (1991). A Simple Weight Decay Can Improve Generalization. page 8.

[80] Langtimm, C. A., Beck, C. A., Edwards, H. H., Fick-Child, K. J., Ackerman, B. B., Barton, S. L., and Hartley, W. C. (2004). Survival Estimates for Florida Manatees from the Photo-Identification of Individuals. *Marine Mammal Science*, 20(3):438–463. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1748-7692.2004.tb01171.x.

[81] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[82] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

[83] Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference Target Propagation. In Appice, A., Rodrigues, P. P., Santos Costa, V., Soares, C., Gama, J., and Jorge, A., editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 498–515. Springer International Publishing.

[84] Lee, H.-S. and Shin, B.-S. (2020). Potato Detection and Segmentation Based on Mask R-CNN. *Journal of Biosystems Engineering*.

[85] Lee, Y.-C., Hsu, H.-W., Ding, J.-J., Hou, W., Chou, L.-S., and Chang, R. Y. (2020). Backbone Alignment and Cascade Tiny Object Detecting Techniques for Dolphin Detection and Classification. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, advpub.

[86] Liang, J. (2018). Google's AI Helps Researcher ID Dolphins.

[87] Liao, Q., Leibo, J. Z., and Poggio, T. (2016). How important is weight symmetry in backpropagation?

[88] Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2014). Random feedback weights support learning in deep neural networks. *arXiv:1411.0247 [cs, q-bio]*. arXiv: 1411.0247.

[89] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2014). Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*. arXiv: 1405.0312.

[90] Linnainmaa, S. (1970). The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7.

[91] Liu, J., Kanazawa, A., Jacobs, D., and Belhumeur, P. (2012). Dog breed classification using part localization. In *European conference on computer vision*, pages 172–185. Springer.

[92] Liu, M., Dong, J., Dong, X., Yu, H., and Qi, L. (2018). Segmentation of Lung Nodule in CT Images Based on Mask R-CNN. In *2018 9th International Conference on Awareness Science and Technology (iCAST)*, pages 1–6. ISSN: 2325-5994.

[93] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *arXiv:1512.02325 [cs]*, 9905:21–37. arXiv: 1512.02325.

[94] Lockyer, C. and Morris, R. (1990). Some observations on wound healing and persistence of scars in Tursiops truncatus. *Reports of the International Whaling Commission*, (12):113–118.

[95] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*. arXiv: 1411.4038.

[96] Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv:1608.03983 [cs, math]*. arXiv: 1608.03983.

[97] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

[98] Maglietta, R., Renò, V., Cipriano, G., Fanizza, C., Milella, A., Stella, E., and Carlucci, R. (2018). DolFin: an innovative digital platform for studying Risso's dolphins in the Northern Ionian Sea (North-eastern Central Mediterranean). *Scientific Reports*, 8(1):17185. Number: 1 Publisher: Nature Publishing Group.

[99] Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. (2013). Fine-Grained Visual Classification of Aircraft. *arXiv:1306.5151 [cs]*. arXiv: 1306.5151.

[100] Mann, J. (2019). Mann-Urian-Google Cloud AI.pdf.

[101] Mann, J., Connor, R. C., Tyack, P., and Whitehead, H. (2000). *Cetacean Societies: Field Studies of Dolphins and Whales*. University of Chicago Press.

[102] Mariani, M., Miragliuolo, A., Mussi, B., Russo, G. F., Ardizzone, G., and Pace, D. S. (2016). Analysis of the natural markings of Risso's dolphins (Grampus griseus) in the central Mediterranean Sea. *Journal of Mammalogy*, 97(6):1512–1524.

[103] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[104] Miragliuolo, A., Mussi, B., and Bearzi, G. (2004). Risso's dolphin harassment by pleasure boaters off the island of Ischia, central Mediterranean Sea. *European Research on Cetaceans*, 15:168–171.

[105] Moore, S. E. (2008). Marine mammals as ecosystem sentinels. *Journal of Mammalogy*, 89(3):534–540.

[106] Morteo, E., Rocha-Olivares, A., Morteo, R., and Weller, D. W. (2017). Phenotypic variation in dorsal fin morphology of coastal bottlenose dolphins (Tursiops truncatus) off Mexico. *PeerJ*, 5:e3415. Publisher: PeerJ Inc.

[107] Neven, D., De Brabandere, B., Proesmans, M., and Van Gool, L. (2019). Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth. *arXiv:1906.11109 [cs]*. arXiv: 1906.11109.

[108] Nguyen, D., Le, T., Tran, T., Vu, H., Le, T., and Doan, H. (2018). Hand segmentation under different viewpoints by combination of Mask R-CNN with tracking. In *2018 5th Asian Conference on Defense Technology (ACDT)*, pages 14–20.

[109] Nie, S., Jiang, Z., Zhang, H., Cai, B., and Yao, Y. (2018). Inshore Ship Detection Based on Mask R-CNN. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 693–696. ISSN: 2153-7003.

[110] Nita, C. and Vandewal, M. (2020). CNN-based object detection and segmentation for maritime domain awareness. In Dijk, J., editor, *Artificial Intelligence and Machine Learning in Defense Applications II*, page 4, Online Only, United Kingdom. SPIE.

[111] Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jojic, N., and Clune, J. (2019). A deep active learning system for species identification and counting in camera trap images. *arXiv:1910.09716 [cs, eess, stat]*. arXiv: 1910.09716.

[112] Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., and 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. Publisher: National Academy of Sciences Section: PNAS Plus.

[113] Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. pages 1037–1045.

[114] of Electrical and Electronics Engineers, I. and Society, I. C., editors (2009). *2009 IEEE Conference on Computer Vision and Pattern Recognition: CVPR 2009 ; Miami [Beach], Florida, USA, 20 - 25 June 2009*. IEEE, Piscataway, NJ.

[115] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[116] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. page 4.

[117] Payne, R. (1986). Long term behavioral studies of the southern right whale (Eubalaena australis). Technical Report 10.

[118] Perrin, W., Würsig, B., and Thewissen, J. (2009). *Encyclopedia of marine mammals*. Academic Press.

[119] Pobar, M. and Ivašić-Kos, M. (2019). Detection of the leading player in handball scenes using Mask R-CNN and STIPS. In *Eleventh International Conference on Machine Vision (ICMV 2018)*, volume 11041, page 110411V. International Society for Optics and Photonics.

[120] Pomeroy, P., O'Connor, L., and Davies, P. (2015). Assessing use of and reaction to unmanned aerial systems in gray and harbor seals during breeding and molt in the UK1. *Journal of Unmanned Vehicle Systems*. Publisher: NRC Research Press http://www.nrcresearchpress.com.

[121] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151.

[122] Qiao, Y., Truman, M., and Sukkarieh, S. (2019). Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming. *Computers and Electronics in Agriculture*, 165:104958.

[123] Quiñonez, Y., Zatarain, O., Lizarraga, C., and Peraza, J. (2019). Using Convolutional Neural Networks to Recognition of Dolphin Images. In Mejia, J., Muñoz, M., Rocha, A., Peña, A., and Pérez-Cisneros, M., editors, *Trends and Applications in Software Engineering*, pages 236–245. Springer International Publishing.

[124] Ramos, E. A., Maloney, B., Magnasco, M. O., and Reiss, D. (2018). Bottlenose Dolphins and Antillean Manatees Respond to Small Multi-Rotor Unmanned Aerial Systems. *Frontiers in Marine Science*, 5. Publisher: Frontiers.

[125] Reddi, S. J., Kale, S., and Kumar, S. (2019). On the Convergence of Adam and Beyond. *arXiv:1904.09237 [cs, math, stat]*. arXiv: 1904.09237.

[126] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Las Vegas, NV, USA. IEEE.

[127] Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*. arXiv: 1804.02767.

[128] Reisser, J., Proietti, M., Kinas, P., and Sazima, I. (2008). Photographic identification of sea turtles: method description and validation, with an estimation of tag loss. *Endangered Species Research*, 5(1):73–82.

[129] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv:1506.01497 [cs]*. arXiv: 1506.01497.

[130] Renò, V., Dimauro, G., Labate, G., Stella, E., Fanizza, C., Cipriano, G., Carlucci, R., and Maglietta, R. (2019). A SIFT-based software system for the photo-identification of the Risso's dolphin. *Ecological Informatics*, 50:95–101.

[131] Riaz, H. u. M., Benbarka, N., and Zell, A. (2020). FourierNet: Compact mask representation for instance segmentation using differentiable shape decoders. *arXiv:2002.02709 [cs, eess]*. arXiv: 2002.02709.

[132] Rohit Malhotra, K., Davoudi, A., Siegel, S., Bihorac, A., and Rashidi, P. (2018). Autonomous Detection of Disruptions in the Intensive Care Unit Using Deep Mask R-CNN. pages 1863–1865.

[133] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*. arXiv: 1505.04597.

[134] Rosso, M., Moulins, A., and Würtz, M. (2008). Colour patterns and pigmentation variability on striped dolphin Stenella coeruleoalba in north-western Mediterranean Sea. *Marine Biological Association of the United Kingdom. Journal of the Marine Biological Association of the United Kingdom*, 88(6):1211–1219. Num Pages: 9 Place: Cambridge, United Kingdom Publisher: Cambridge University Press.

[135] Rother, C., Kolmogorov, V., and Blake, A. (2004). "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts. *ACM transactions on graphics*, 23(3):309–314.

[136] Rowcliffe, J. M., Field, J., Turvey, S. T., and Carbone, C. (2008). Estimating Animal Density Using Camera Traps without the Need for Individual Recognition. *Journal of Applied Ecology*, 45(4):1228–1236. Publisher: [British Ecological Society, Wiley].

[137] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*. arXiv: 1609.04747.

[138] Rumelhart, D. E., Hintont, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. page 4.

[139] Schevill, W. E. and Backus, R. H. (1960). Daily Patrol of a Megaptera. *Journal of Mammalogy*, 41(2):279.

[140] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. pages 815–823.

[141] Sharma, P. (2019). Image Segmentation Python | Implementation of Mask R-CNN | https://analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/.

[142] Sharpe, M. and Berggren, P. (2019). Indian Ocean humpback dolphin in the Menai Bay off the south coast of Zanzibar, East Africa is Critically Endangered. *Aquatic Conservation: Marine and Freshwater Ecosystems*, 29(12):2133–2146. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aqc.3221.

[143] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*. arXiv: 1409.1556 version: 6.

[144] Society, W. M. (1970). The Beaufort Scale of Wind Force:(Technical and Operational Aspects). 3.

[145] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:30.

[146] Swanson, A., Kosmala, M., Lintott, C., Simpson, R., Smith, A., and Packer, C. (2015). Snapshot Serengeti, high-frequency annotated camera trap images of 40 mammalian species in an African savanna. *Scientific Data*, 2(1):150026. Number: 1 Publisher: Nature Publishing Group.

[147] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going Deeper With Convolutions. pages 1–9.

[148] Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., Vercauteren, K. C., Snow, N. P., Halseth, J. M., Salvo, P. A. D., Lewis, J. S., White, M. D., Teton, B., Beasley, J. C., Schlichting, P. E., Boughton, R. K., Wight, B., Newkirk, E. S., Ivan, J. S., Odell, E. A., Brook, R. K., Lukacs, P. M., Moeller, A. K., Mandeville, E. G., Clune, J., and Miller, R. S. (2019). Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, 10(4):585–590. _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13120.

[149] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification.

[150] Tan, M., Pang, R., and Le, Q. V. (2020). EfficientDet: Scalable and Efficient Object Detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, Seattle, WA, USA. IEEE.

[151] Thompson, J. (2019). finFindR.pdf.

[152] Tian, Z., Shen, C., Chen, H., and He, T. (2019). FCOS: Fully Convolutional One-Stage Object Detection. *arXiv:1904.01355 [cs]*. arXiv: 1904.01355.

[153] Tieleman, T. and Hinton, G. (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.

[154] Trotter, C., Atkinson, G., Sharpe, M., Richardson, K., McGough, A. S., Wright, N., Burville, B., and Berggren, P. (2020). NDD20: A large-scale few-shot dolphin dataset for coarse and fine-grained categorisation. *arXiv:2005.13359 [cs]*. arXiv: 2005.13359.

[155] Tzutalin (2021). LabelImg. original-date: 2015-09-17T01:33:59Z.

[156] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

[157] University, G. (2018). Is That 'Jimmy Carter' or 'Barbara Bush?' Google Designs, Professor Uses Artificial Intelligence to Track Wildlife.

[158] Urian, K., Gorgone, A., Read, A., Balmer, B., Wells, R. S., Berggren, P., Durban, J., Eguchi, T., Rayment, W., and Hammond, P. S. (2015). Recommendations for photo-identification methods used in capture-recapture models with cetaceans. *Marine Mammal Science*, 31(1):298–321.

[159] Van Bressem, M.-F., Burville, B., Sharpe, M., Berggren, P., and Van Waerebeek, K. (2018). Visual health assessment of white-beaked dolphins off the coast of Northumberland, North Sea, using underwater photography. *Marine Mammal Science*, 34(4):1119–1133.

[160] Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8769–8778.

[161] VanBressem, M.-F., Burville, B., Sharpe, M., Berggren, P., and VanWaerebeek, K. (2018). Visual health assessment of white-beaked dolphins off the coast of Northumberland, North Sea, using underwater photography. *Marine Mammal Science*.

[162] Vernazzani, R. G. and Cabrera, E. (2013). Eastern South Pacific southern right whale photoidentification catalog reveals behavior and habitat use patterns. *MARINE MAMMAL SCIENCE*, page 10.

[163] Vuola, A. O., Akram, S. U., and Kannala, J. (2019). Mask-RCNN and U-net Ensembled for Nuclei Segmentation. *arXiv:1901.10170 [cs]*. arXiv: 1901.10170.

[164] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The Caltech-UCSD Birds-200-2011 Dataset - CaltechAUTHORS.

[165] Waleed, A. (2017). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. original-date: 2017-10-19T20:28:34Z.

[166] Wang, X., Kong, T., Shen, C., Jiang, Y., and Li, L. (2020a). SOLO: Segmenting Objects by Locations. *arXiv:1912.04488 [cs]*. arXiv: 1912.04488.

[167] Wang, X., Zhang, R., Kong, T., Li, L., and Shen, C. (2020b). SOLOv2: Dynamic and Fast Instance Segmentation. *arXiv:2003.10152 [cs]*. arXiv: 2003.10152.

[168] Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244.

[169] Weideman, H. J., Jablons, Z. M., Holmberg, J., Flynn, K., Calambokidis, J., Tyson, R. B., Allen, J. B., Wells, R. S., Hupman, K., Urian, K., and Stewart, C. V. (2017). Integral Curvature Representation and Matching Algorithms for Identification of Dolphins and Whales. *arXiv:1708.07785 [cs]*. arXiv: 1708.07785.

[170] Weinstein, B. G. (2018). A computer vision for animal ecology. *Journal of Animal Ecology*, 87(3):533–545.

[171] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200.

[172] Weller, D. (1998). *Global and regional variation in the biology and behavior of bottlenose dolphins.* PhD thesis, Texas A&M University, College Station.

[173] Willi, M., Pitman, R. T., Cardoso, A. W., Locke, C., Swanson, A., Boyer, A., Veldthuis, M., and Fortson, L. (2019). Identifying animal species in camera trap images using deep learning and citizen science. *Methods in Ecology and Evolution*, 10(1):80–91. _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13099.

[174] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2020). Detectron2. original-date: 2019-09-05T21:30:20Z.

[175] Würsig, B. and Jefferson, T. A. (1990). Methods of photo-identification for small cetaceans. *Reports of the International Whaling Commission*, 12:43–52.

[176] Würsig, B. and Würsig, M. (1977). The Photographic Determination of Group Size, Composition, and Stability of Coastal Porpoises (Tursiops truncatus). *Science*, 198(4318):755–756.

[177] Xie, E., Sun, P., Song, X., Wang, W., Liang, D., Shen, C., and Luo, P. (2020). PolarMask: Single Shot Instance Segmentation with Polar Representation. *arXiv:1909.13226 [cs]*. arXiv: 1909.13226.

[178] Xie, L., Tian, Q., Hong, R., Yan, S., and Zhang, B. (2013). Hierarchical Part Matching for Fine-Grained Visual Categorization. In *2013 IEEE International Conference on Computer Vision*, pages 1641–1648, Sydney, Australia. IEEE.

[179] Xu, W., Wang, H., Qi, F., and Lu, C. (2019). Explicit Shape Encoding for Real-Time Instance Segmentation. *arXiv:1908.04067 [cs]*. arXiv: 1908.04067.

[180] Yu, Y., Zhang, K., Yang, L., and Zhang, D. (2019). Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Computers and Electronics in Agriculture*, 163:104846.

[181] Zeiler, M. D. and Fergus, R. (2013). Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. *arXiv:1301.3557 [cs, stat]*. arXiv: 1301.3557.

[182] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). Mixup: Beyond Empirical Risk Minimization. *arXiv:1710.09412 [cs, stat]*. arXiv: 1710.09412.

[183] Zhang, N., Donahue, J., Girshick, R., and Darrell, T. (2014). Part-Based R-CNNs for Fine-Grained Category Detection. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 834–849. Springer International Publishing.

[184] Zhang, N., Farrell, R., Iandola, F., and Darrell, T. (2013). Deformable part descriptors for fine-grained recognition and attribute prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 729–736.

[185] Zhao, T., Yang, Y., Niu, H., Wang, D., and Chen, Y. (2018). Comparing U-Net convolutional network with mask R-CNN in the performances of pomegranate tree canopy segmentation. In *Multispectral, Hyperspectral, and Ultraspectral Remote Sensing Technology, Techniques and Applications VII*, volume 10780, page 107801J. International Society for Optics and Photonics.

[186] Zhou, X., Zhuo, J., and Krähenbühl, P. (2019). Bottom-up Object Detection by Grouping Extreme and Center Points. *arXiv:1901.08043 [cs]*. arXiv: 1901.08043.