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

Acknowledgements

ACKNOWLEDGEMENTS

Abstract

ABSTRACT

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

CNN Convolutional Neural Networks

CPU Central Processing Unit

CV Computer Vision

FCN Fully Convolutional Network

GPU Graphical Processing Unit

MCZ Marine Conservation Zone

NDD20 Northumberland Dolphin Dataset 2020

PCA Principle Component Analysis

ReLU Rectified Linear Unit

RIB Rigid Inflatable Boat

RPN Region Proposal Network

SGDR Stochastic Gradient Descent with Restarts

SGD Stochastic Gradient Descent

SIFT Speeded-Up Robust Features

SMRU Sea Mammal Research Unit

SURF Speeded-Up Robust Features

TPE Tree-structured Parzen Estimator

VM Virtual Machine

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 [149]. 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 [52, 238]. 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.

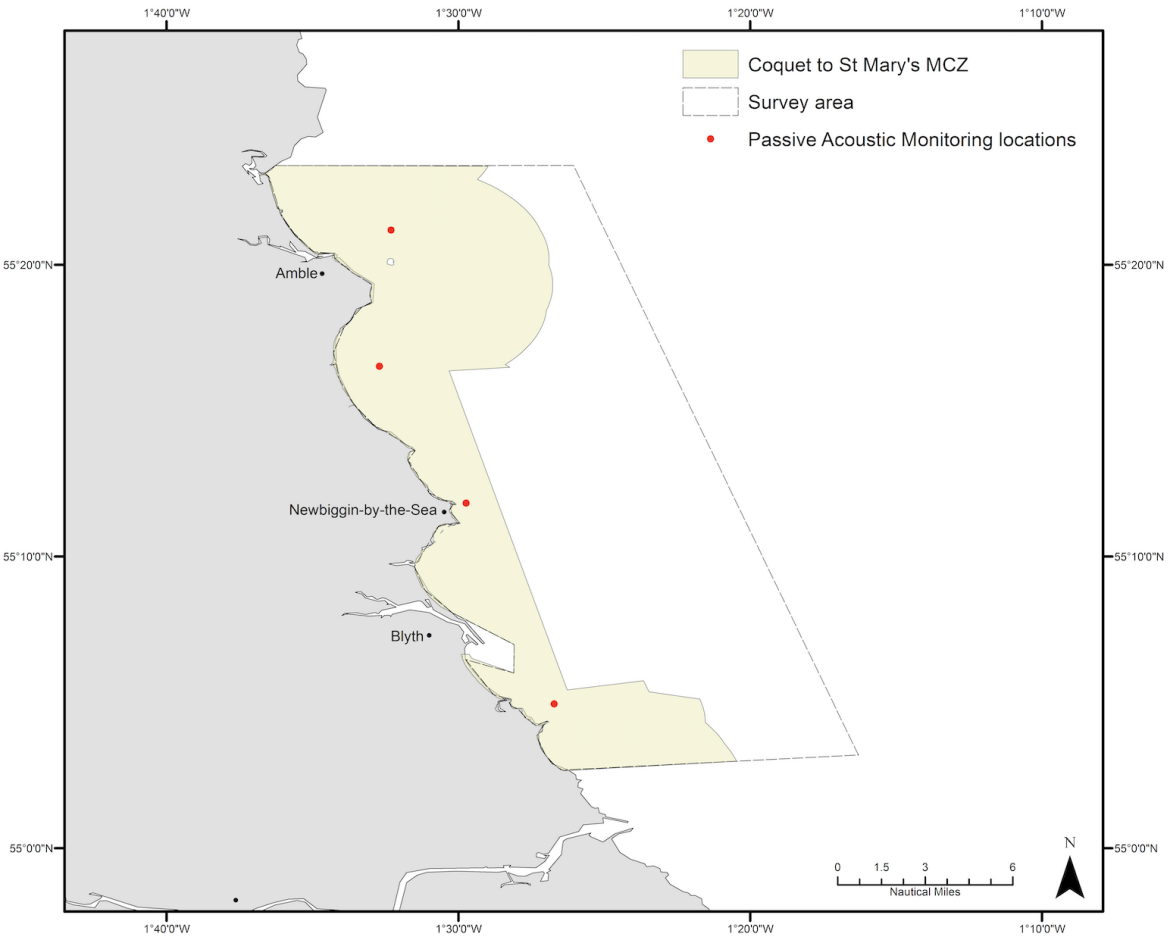

Figure 1.1 Map of the survey area, Northumberland, UK, from St. Mary's Lighthouse in the south to 25nm above Coquet Island in the north.

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 [74, 88] 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 [216]. 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 to tackle problems faced in an ever increasing range of areas. Deep learning models, large neural networks capable of exploiting abstract patterns in data to solve a task, have consistently been shown to outperform other machine learning techniques [114, 121, 162, 250]. As an extremely fast paced and ever growing field it would not be possible to explore the entirety of deep learning, and as such this Chapter will focus primarily on deep learning in a computer vision context, exploring and understanding image data.

One novel area where computer vision and deep learning can play an important role is in the world of marine conservation, helping to automate a currently labour intensive discipline. This project focusses on the automation of cetacean photo-identification, a process utilised by conservationists for tasks such as population estimates and health assessments [45, 100, 134, 216]. Before the research undertaken in this project is discussed, this Chapter will seek to provide an introduction to both photo-identification and deep learning, before expanding into how this has been applied to computer vision. Literature focussing on computer vision in a cetacean conservation space is explored, as well as the current state of fine-grained recognition - utilising computer vision algorithms to differentiate between visually similar classes.

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 [28, 53, 188]. 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 can be 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 [64, 89], 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 [69, 163, 237]. 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 [46, 78].

Initially utilised for the tracking of individual distinctive animals within a species [40, 184], the methodology was quickly adapted to large-scale monitoring of whole pods [6, 70]. Photo-id has been utilised for the monitoring of multiple cetacean species, with proven use cases in a range of studies such as those focussing on Indian Ocean humpback dolphins (*Sousa plumbea*) [188], Risso's dolphins (*Grampus griseus*) [147], Northern bottlenose whales (*Hyperoodon ampullatus*) [65], and killer whales (*Orcinus orca*) [28]. Outside of cetaceans, photo-id has further found use studying other marine life such as whale sharks (*Rhincodon typus*) [100], sea turtles (both *Chelonia mydas* and *Eretmochelys imbricata*) [174], and Florida manatees (*Trichechus manatus latirostris*) [119]. Land based photo-identification studies are also possible, with Goswami *et al*. utilising photographic data to estimate demographic parameters of Asian elephants (*Elephas maximus*) [83].

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 - examples of underwater photo-id do exist however the practise is not yet commonplace [218, 223]. 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 [12, 15, 53, 188, 219]. 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 [134, 144, 238]. 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

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*) [164]. Right: fluke of a humpback whale (*Megaptera novaeangliae*) [42]. Bottom: dorsal fin of a common bottlenose dolphin (*Tursiops truncatus*) [211].

with a vessel, or natural, for example from encounters with prey. Examples of prominent markings used in dolphin photo-identification surveys can be seen in Figure 2.2.

Scarring is of particular use when identifying Risso's dolphins who are well known for the persistent nature of their scars, which is thought to occur due to a loss of pigmentation when their scars heal [145]. 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 [180].

Regardless of the species being analysed or the body-parts used during photo-id it is imperative that the process is standardised, allowing for work to be compared over spatial and temporal scales. This process began in 1988 through workshops held by the International Whaling Commission, with further recommendations published in 2015 by Urian *et al.* [89, 215].

This standardisation process requires some assumptions to be universally made. For one, all of the markers must be 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

Figure 2.2 Examples of prominent markings used during dolphin photo-identification surveys. Top Left: nicks. Top Right: scarring. Bottom Left: scratches. Bottom Right: pigmentation.

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 provide 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.

Because of the assumptions which must be adhered to, as well as the manual nature of the photo-id process, there are many downsides to the process. 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. If the individual has a very distinct fin shape then this issue can be overcome, although this may not always be possible. As such, individuals may not be added to a catalogue without both sides of the fin available for later comparison.

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 vessels. 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 analysed 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 Machine Learning: Supervised vs Unsupervised Approaches

Before it is possible to understand how deep learning and computer vision, both subfields of machine learning, can be utilised to aid in the photo-id procees, it is important to discuss the differences between supervised and unsupervised machine learning.

Supervised learning tasks are those where the model can be trained using an input and expected output value pair, known as a ground truth. This technique lends itself well to tasks such as classification, where an input can be mapped to a set of defined output classes, or in regression where an input can be mapped to a continuous output space.

Training 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; for example 80% of the data used for training with 20% used for testing. The classifier will then iterate through the training set, using the ground truth values to train the network's parameters in a way to best generalise the model. After training has been completed the model will then be evaluated against the test set. Each data point will be processed by the model and a prediction outputted, which is then compared to the unseen ground truth to provide an evaluation of the model's performance.

Unsupervised learning tasks are, in contrast, those where prior ground truths for your data are not available. This approach lends itself well to clustering and aiding in understanding of the underlying data structure. These unsupervised algorithms, such as K-Means clustering [92], are not provided human guidance on how to group the data given, but are rather left to discover interesting structure patterns on their own.

Taking the `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. However due to the unsupervised nature of the learning process, the model is equally as likely to cluster the data based on whether the animal is, for example, sitting or standing.

2.3 A Brief Introduction to Deep Learning

Deep learning, a subfield of machine learning, aims to create artificial networks to complete tasks through a learning process, in a similar way to how the human brain operates. These computational models are made up of neurons and are often multiple layers deep. Lower layers represent basic abstractions building up from this as you go *deeper* into the network. Layers at the deepest points can, based on information passed to them from lower levels, begin to provide estimations of answers to a given problem. In literature neural networks are often illustrated in forms similar to Figure 2.3, which shows a visual representation of the VGG16 architecture [192].

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. This can be seen as deep learning democratising machine learning thanks to the model learning purely from the data rather than any bias added in by humans which may be present using non-deep learning approaches.

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. For example in the case of computer vision, lower layers of neurons are optimised by the network itself 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

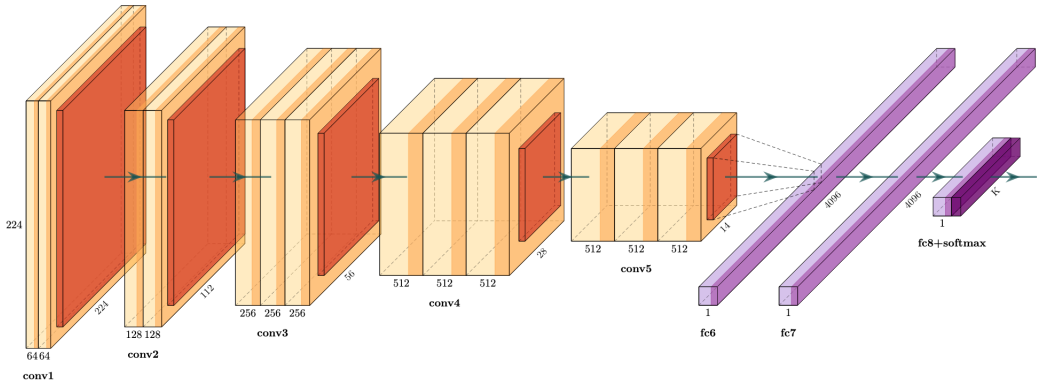

Figure 2.3 An example neural network architecture diagram, showing VGG16 [192]. Data is inputted to the network at the leftmost layer before being passed through the layers sequentially. The rightmost layer provides the overall output of the network, such as a classification or probability. Image from [91].

updating their parameters predominantly through optimisations such as stochastic gradient descent and backpropagation [96].

This ambition to create artificial networks similar to how the brain operates stems mainly from work undertaken in 1943 by McCulloch and Pitts [146] 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 of large scale datasets needed to train these networks and power of the computing resources available has deep learning research accelerated. The transition to training on clusters of high-powered GPUs has allowed for a significant speed-up in both model training and inference time compared to traditional CPUs, allowing for a higher amount of prototyping in a smaller time frame [138]. 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 have reduced the need for researchers to acquire their own hardware for model training - although many argue it is still cheaper to buy your own in the long term.

More so, advances in deep learning have been helped greatly through the development of standard programming frameworks. Google's Tensorflow [2] and Facebook's PyTorch [161] 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 large scale coarse-grained datasets such as MNIST [122], Caltech-256 [85], and ImageNet

[57] 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 [160]. 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 [196], batch normalisation [104], stochastic gradient descent with warm restarts [136], and mixup [249]. The use of data augmentation is also commonplace. This is where existing data is perturbed randomly to create new, artificial data. Examples of image data augmentation include simple perturbations such as random horizontal and vertical flipping, and cropping, as well as more complicated techniques like Gaussian blurring, perspective shifting, and colour jitter.

2.3.1 Optimising Deep Learning Networks

In order to generalise our deep learning models, we need to be able to optimise the parameters within each individual neuron. These parameters can either be weights which control the strength of the connection between two neurons, or biases - a constant additional input which guarantees a neuron's value can never be zero. Most commonly, this is performed using gradient descent to minimise a loss function (a measure of distance between ground truth and model prediction). If this function were to be visualised on a graph each parameter would be represented by an axes, resulting in a hyper-surface with millions of dimensions in the case of deep learning models. The goal of network optimisation is to find the minimum point on the hyper-surface, as this would give the parameter values which produce the smallest loss.

Loss functions however are non-convex [48], which can result in multiple local minimums. In order to find the (hopefully global) minimum point on the hyper-surface, during training the model's weights must be updated iteratively in the *opposite* direction of the gradient of the loss function's hyper-surface. As such, we follow the direction of the slope of the hyper-surface downhill until we reach a minima, an area where the loss is lowest [182].

Before the advent of deep learning and big data, it was common for the whole training set to be used to compute the gradient at each iteration; however thanks 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, a small random subset of the larger dataset, are often used to give an estimation of the overall loss gradient.

In order to achieve this, a process known as stochastic gradient descent (SGD) is 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 non-batch 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 [32].

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 (SGDR) [136], Momentum [167], RMSProp [209], Adam [113], and AMSGrad [171]. Work in this thesis utilises SGDR and Adam primarily. 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 [48, 56].

2.3.2 Backpropagation

As discussed in Section 2.3.1, 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 [5].

Backpropagation was originally described in Linnainmaa's masters thesis from 1970 [130], although its effect was not fully realised until 1986, when Rumelhart *et al.* discussed the advantages to using backpropagation over other learning approaches [183]. In recent years multiple works have provided updates and improvements to the original backpropagation algorithm, however none have seen wide-scale adoption [22, 123, 127, 128, 157].

The standard backpropagation algorithm to compute the gradient of the loss function with respect to 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.

2.4 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 [221]. Generalisable concepts such as Convolutional Neural Networks (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 its 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; 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 which have cyan, magenta, yellow, and black colour channels would have a depth of four. As can be imagined, these matrices can very quickly reach unworkable sizes. An RGB image at 1080p resolution for example would require a matrix of size 1920x1080x3, or 6,220,800 values. CNNs utilise a set of standard operations which are capable of reducing image sizes down to a more workable form whilst still retaining key features which allow for inference to occur.

2.4.1 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

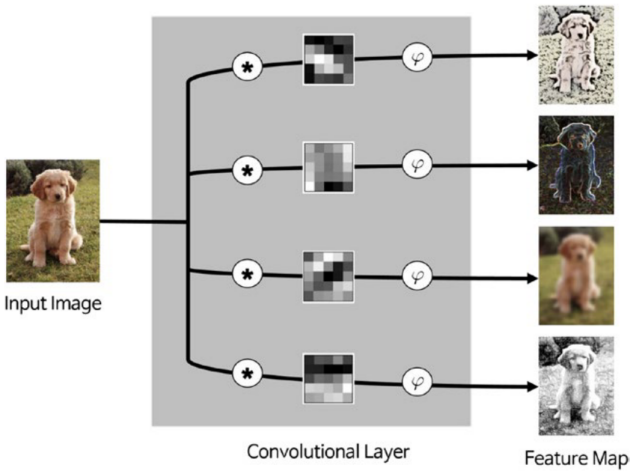

Figure 2.4 A visual representation of convolution. An image, left, is fed into a convolutional layer. The input is passed through a convolution operation, $*$. The greyscale blocks in the centre of the convolutional layer represent the kernels passed over the image. φ represents the activation function the kernel output is fed to. This process results in multiple outputted feature maps. Image from [112].

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 and passing them through an activation function. 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 then built on as the model gets deeper [112]. The kernel process can be performed multiple times over the same input using different weightings, giving multiple output feature maps. A visual representation of a convolutional layer can be seen in Figure 2.4.

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 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. Figure 2.5 shows a visualisation of the kernel process with a stride of 2.

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 can

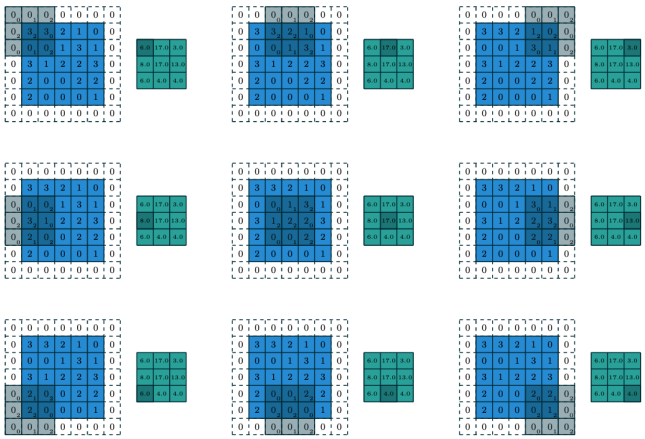

Figure 2.5 A kernel, represented by the grey squares, operates over a padded input matrix, blue, with a stride of 2 to produce an output feature map, green. Note that the kernel's weights are denoted by the number in the lower right of each box and the input pixel value is denoted by the number in the middle of each box. Image from [61].

never be in the centre of the kernel. This can cause a 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, zero padding can be performed [4]. This technique places zeros around the edges of the input matrix, expanding its size and allowing pixel values formerly at the edges to be utilised by the kernel. This also prevents the size of the input shrinking between multiple convolution operations, allowing for deep networks to operate on small input images.

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 applied over batches of the input pixel values, similar to how a kernel operates in a convolution layer by sliding over the image. 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 which only keeps the maximum pixel value in the batch, average pooling [35] which outputs

the mean of all pixel values in the batch, and stochastic pooling [248] which selects an output pixel value from each batch based on a probability distribution. For a review of current pooling methods, see [77].

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. The activation function is responsible for deciding which neuron in the layer passes its value to the layer below by computing the weighted sum of the inputs and passing the result through a non-linear function. Softmax 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 [116] or as a category for classification as the last layer of the network [82].

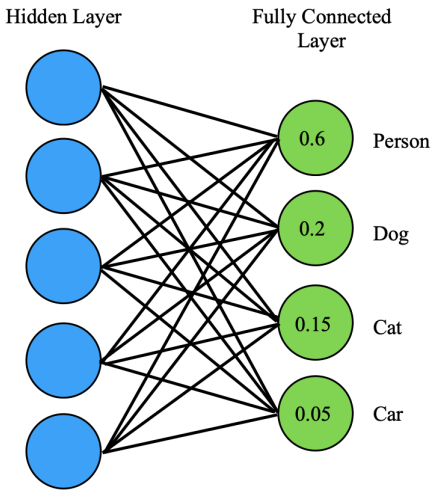

Figure 2.6 A visual representation of the final stages of a classification network. The leftmost blue circles represent the final hidden layer in the network, with the rightmost green circles representing the fully connected layer. Each neuron in the hidden layer is connected to all neurons in the fully connected layer. The value in the hidden layer's neurons represents the activations after softmaxing, with the label denoting the class represented by the neuron. The network has predicted the input to be of class `Person`.

For example let's assume we have a network who's aim is to classify an image into one of four classes, as seen in Figure 2.6. At the end of this network is a fully connected layer which takes a feature map from the proceeding hidden layer, and produces as output four 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 four neurons where each neuron represents a possible class for the input 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 an infinite number of CNN architectures. 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). 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 [63], 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 [122] 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 [116]. Utilising a similar but deeper architecture to LeNet, with more filters and a larger number of stacked convolutional layers, AlexNet also included now commonplace deep learning building blocks such as dropout [196], max pooling [35], and ReLU activation functions; the most popular non-linear activation function currently in deep learning, especially in computer vision [95]. 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 2.7. 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 [199]. This net achieved a top-5 error rate of 6.67%, very close to what untrained humans could achieve on the competition dataset, ImageNet [116]. 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 up to 152 layers deep, and achieved a human-beating top-5 error rate of 3.57% [94]. Shallower versions of ResNet exist, such as ResNet50 and ResNet101, which are 50 and 101 layers deep respectively. This thesis makes use of ResNet50.

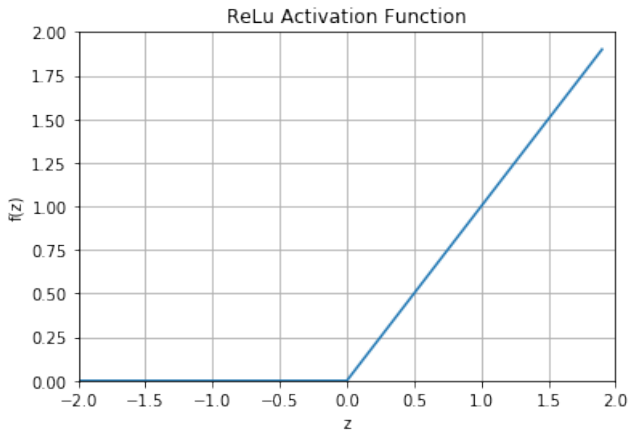

Figure 2.7 A visualisation of the ReLU Activation Function.

2.4.2 Object Detection

Thanks to advancements in deep learning technology and the creation of standardised architectures, CNNs are now utilised en masse to perform tasks such as object detection, attempting 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. One of the most common and widely used RPN architectures is derived from the R-CNN, or Regions with CNN features, architecture [82]. R-CNN utilises a selection search [213] to generate 2000 Regions of Interest (RoIs) representing the most likely areas of the input image to contain a class example. By limiting the number of RoIs generated this allows for fast computation compared to operating on every possible region in the image. The proposed RoIs are then fed through a CNN to be classified. See Figure 2.8 for a visual representation of the R-CNN pipeline. Example proposed RoIs can be seen in Figure 2.9.

Utilising selection search leads to a high recall rate thanks to the large amount of proposals, as there is a high probability that some of these proposals will contain RoIs with 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

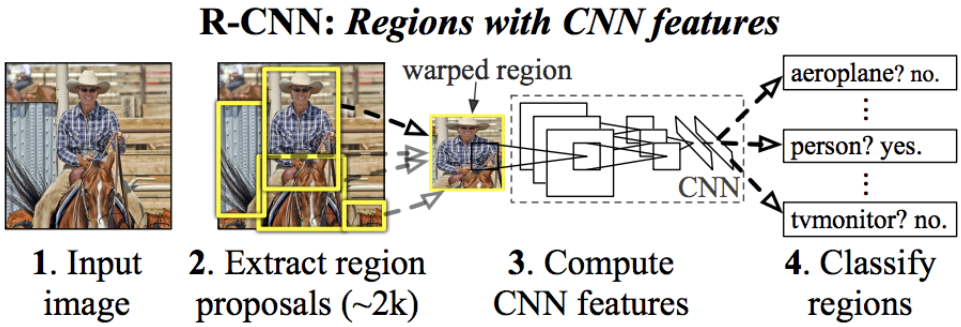

Figure 2.8 The R-CNN pipeline. Image from [82].

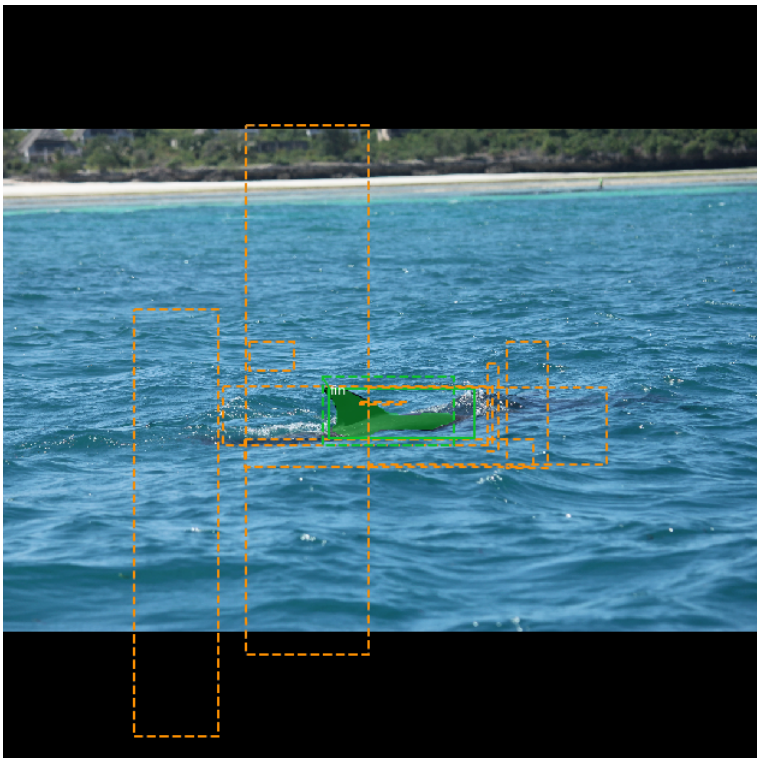

Figure 2.9 An example of RoIs generated on an image by an RPN, showing 10 random proposals. Note that two of the RoIs have been successfully classified as `fin`.

the network needs to classify the 2000 region proposals generated. Detection can also be slow using an R-CNN and, with the selection search being fixed, no adaptive learning takes place here which may lead to bad region proposal generation.

Some of these time drawbacks were fixed in a later version of R-CNN, known as Fast-RCNN [81]. 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 for backpropagation to occur throughout the network, greatly improving run times. This also means however that the runtime is determined by how fast RoIs can be generated.

To fix this issue, Faster R-CNN was developed [175]. 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 [172] and SSD [133].

2.4.3 Semantic Segmentation

Along with object detection, semantic segmentation is one of the key research areas in computer vision. Rather than provide RoI bounding boxes as output, semantic segmenters instead output a class label for each pixel in the image. A group of connected pixels of the same class is known as a mask. An example of an image and its masks can be seen in Figure 2.10.

Figure 2.10 An example of an image and its corresponding ground truth `fin` masks.

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 operating over multiple proposal boxes, known as anchors, 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. An example of generated negative, neutral, and positive anchors for an image can be seen in Figure 2.11.

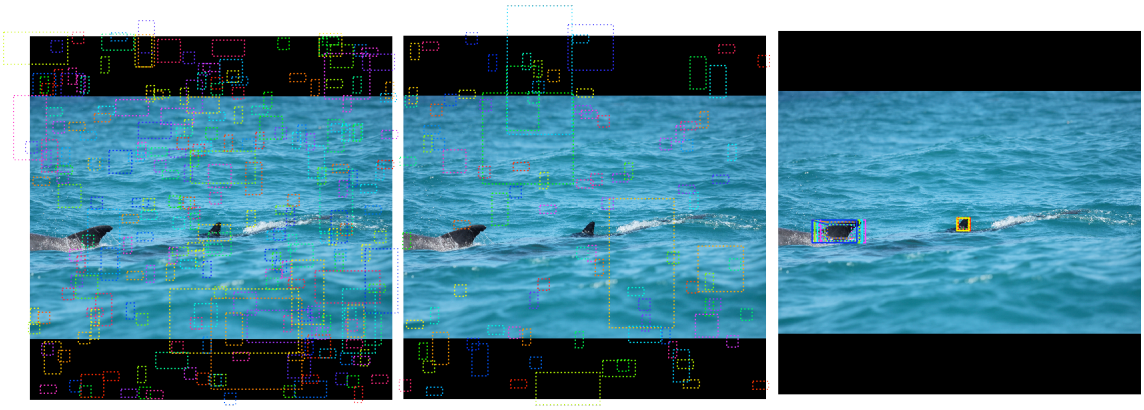

Figure 2.11 Generated anchors. Left: negative anchors. Middle: neutral anchors. Right: positive anchors.

In some cases, positive anchors may 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 2.12.

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 [90] or Facebook's Mask R-CNN [93]. This thesis makes use of Mask R-CNN.

Fully Convolutional Networks (FCN) can also be utilised for semantic segmentation [135]. An FCN learns pixel to pixel mappings without the need for region proposals and are built using only 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 been tackled through the development of more advanced FCNs such as SegNet [14] and DeepLab [44].

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 [37]. However doing this can lead to a wide variance in the quality of ground truth masks generated due to the financial incentive for those creating the masks to work as quickly as possible, necessitating the need to develop bespoke systems for quality control [140].

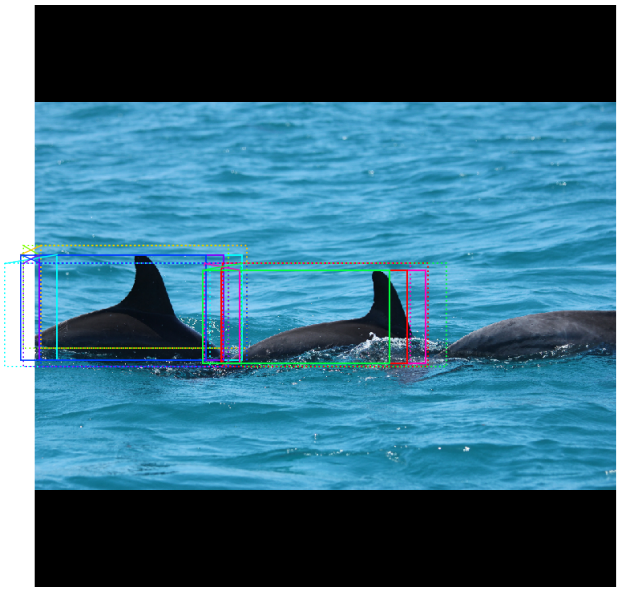

Figure 2.12 An example of refined anchors. Positive anchors before refinement are dotted, after refinement are solid.

2.4.4 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 [251]. This is still an active area of research, with some approaches focusing on a form of hierarchical part matching [240], some on alignment of objects to define a super-class shape [76], some utilising deformable part descriptors [252], and others using part localisation [131].

At first glance it may seem as though part segmentation would be useful for this thesis' work into automatic photo-identification. However, when analysing the target data it becomes apparent there would be little benefit to this approach over other segmentation techniques. As this work focusses on above water photo-identification the vast majority of images observed by the developed system only contain one part of the animal - the dorsal fin as it breaches the waterline. As such there would be little advantage to the use of part segmentation here, as only one part is visible. If this work shifted focus to include underwater photo-id, part segmentation would be extremely useful. In this case there would be multiple parts of the animal visible such as the dorsal, tail stock, head, ventral, or sides. Each of these parts would have their own prominent markings useful for identification. However even if all parts of the animal are visible, it may be the case due to water conditions that not all prominent

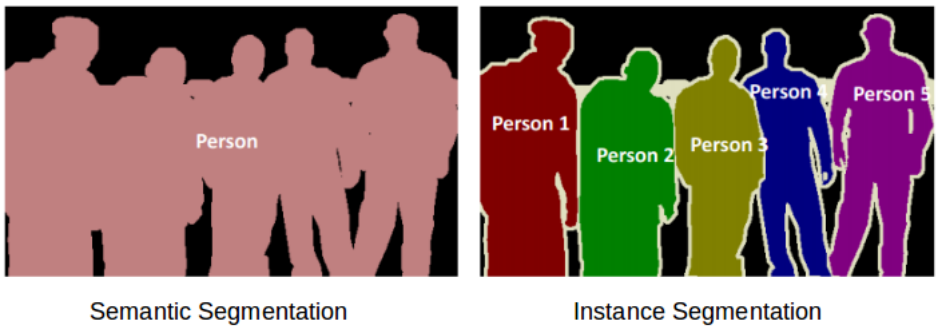

Figure 2.13 An example showing the difference between semantic and instance segmentation. In semantic segmentation, all pixels which belong to the `person` class have been classified as one `person` object. In instance segmentation, five `person` objects have been detected, and all `person` pixels have been assigned to one of the objects. Image from [187].

markings are. Algae bloom and light refraction can obscure an individual's identifying information, and so it would be very useful to be able to break down an individual using part segmentation, allowing for identification to be performed on each part rather than the whole animal, potentially increasing the chance of identification.

2.4.5 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 2.13.

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. These systems are often expensive to develop however due to the increased workload of data labelling required. Compared to data labelled for object detection, which only requires a bounding box ground truth, instance segmented data is required to be labelled on a pixel basis in a similar manner as semantically segmented data. However instance segmented data requires each pixel be assigned a specific object if there are multiple objects of the same class in the image which is not the case for semantically segmented data.

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 pixels inside of the box belong to either the target or background class. Examples of *top-down* approaches include Faster R-CNN [175], and Mask R-CNN [93]. In contrast, *bottom-up* systems first segment then detect, such as SpatialEmbedding [151] 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. 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 [195].

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 struggle to reach levels of segmentation accuracy seen when utilising two stage systems [195]. ExtremeNet [255] 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 [241] utilises the concept of Chebyshev polynomials to fit a radius around each object inside of the detected bounding box. Similarly, PolarMask [239] also represents masks through the use of a contour around the object, modelling this through the use of polar coordinates. FourierNet [177] 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 [31] builds on the well known YOLO object detection architecture, specifically YOLOv3 [173], 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 [43] works in a similar way to YOLOACT however predicts an attention map rather than instance coefficients and utilises FCOS [208] as a backbone, a completely anchor and proposal free object detection architecture resulting in reduced complexity when compared to YOLO [172] and SSD [133].

Whilst the majority of one stage approaches to instance segmentation rely on bounding boxes, this is not always the case. SOLO [227] introduces the concept of instance categories, assigning categories to each pixel according to the size and location of the instance. SOLOv2 [228] 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 [93]. 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 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.* [175] (itself an extension of Fast R-CNN developed in 2015 [81]).

Faster R-CNN is a two stage architecture. The first stage utilises a standard backbone network, typically ResNet [94], VGG [192], or Inception [199], to convert an input image into a set of feature maps which are passed to an RPN for analysis (see Section 2.4.2 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

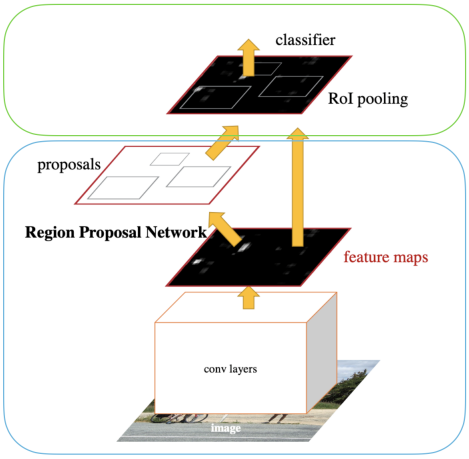

Figure 2.14 The Faster R-CNN architecture [175]. The blue box represents operations in stage one, which includes a standard backbone CNN architecture and RPN. The green box represents operations in stage two, performing RoI pooling and classification.

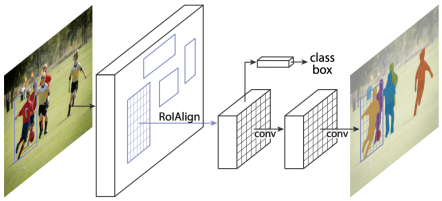

Figure 2.15 A visual representation of the changes made to stage two of Faster R-CNN to create Mask R-CNN. The RoI Pooling layer has been replaced with an RoI Align layer, along with the addition of a mask head. Image from [93].

predict bounding boxes, classifications, and confidence scores. A visual representation of Faster R-CNN's architecture can be seen in Figure 2.14.

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 2.15 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 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 [7, 47, 132, 178], agricultural [50, 124, 168, 253], sports [38, 152, 165], astronomical [39], and nautical [101, 153] fields. Alongside being well known, Mask R-CNN is also extremely reproducible. An official PyTorch implementation is available through Facebook AI Research's Github [236], whilst Matterport's Mask R-CNN implementation is most commonly utilised when working with Tensorflow (including in this project) [225].

2.4.6 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 better than human levels of accuracy on a wide variety of tasks; at the time of writing the current state of the art for ImageNet [158] and CIFAR-10 [115], two of the most commonly used classification benchmark datasets, are both held by Foret *et al.* utilising EfficinetNet with SAM [68] at 88.61% and 99.70% top-1 accuracy respectively. 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 contrast to the coarse-grain nature of the datasets above, fine-grain datasets are those with a small interclass variation. Whilst CIFAR-10 has one class covering all different types of `dog`, the fine-grain dataset Stanford Dogs [111] 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 [233] and the updated Caltech-UCSD Birds 200-2011 [224], iWildcam for camera trap data [20], and FGVC Aircraft [141].

Whilst fine-grain datasets may contain small inter-class variation, their intra-class variation can be large. Class examples may contain a wide variety of orientations, poses, colour, and sizes. This allows for trained models to generalise and be capable of detecting class examples in a wider variety of cases. 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 however ranks 31st in the FGVC Aircraft benchmark ranking on Papers With Code [1] at the time of writing.

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

2.4.7 Extreme Fine-Grained Visual Categorisation

Residing at the end of the scale of classification granularity is the task of extreme fine-grain classification, where the differences between dataset classes are minute. Developing extreme fine-grained datasets is extremely difficult, often requiring the input of domain experts to label class examples. Taking photo-id data as an example, at a coarse-grained level it would be enough to label all classes in a photo-id catalogue as `dolphin`. At a fine-grained level multiple classes may start to exist, such as splitting based on species. Building a useable photo-id catalogue however is an extreme fine-grained task, where classes are split based on the individual.

Often there will be very few prominent markings which will allow for an individual to be classified, and these markings are often very small such as a notch in the fin or a scar; the vast majority of pixels in the classes will be very similar. Because of this, photo-id catalogues can only be accurately produced by local conservationists, those who have studied the resident cetacean population for many years. Even with this expertise however, creating an extreme fine-grained dataset such as a photo-id catalogue can be a large undertaking, often requiring many months of work to ensure all classifications are correct. An example highlighting the differences between coarse, fine, and extreme fine-grained recognition can be seen in Figure 2.16.

2.5 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 [232].

Work into applying object detection and segmentation to conservation data mostly focusses on camera trap systems due to the large amount of data readily available. 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 [198]. The iWildcam dataset provides further camera trap training data from across the South-western United States [20].

[2]Snapshot Serengeti: snapshotserengeti.org

Figure 2.16 An example image from a photo-id catalogue with one individual present. At a coarse-grain level, this individual is classed as `dolphin`, at a fine-grained level `WBD` (white-beaked dolphin), and at an extreme fine-grained level `32`, the individual's catalogue ID. Image and class labels from [211].

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 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 [21, 155, 156, 200, 235]. 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 population, a significant achievement given the challenge of identifying individuals within a species which do not have unique markings [51].

Applying computer vision to marine environments is a greater challenge than on-land camera traps. This is, in part, due to the relative lack of available data to train systems. Camera traps work on movement taking a burst of images whenever the environment they observe changes, be this due to an animal walking through the scene or wind moving foliage. They must also be kept stationary, usually mounted to grounded objects such as trees. These requirements make camera traps unsuitable for marine environments, as the trap could not be attached to a stationary object at sea. Should it be possible to provide a stationary mount for the camera trap the environment would still be unsuitable due to the rapid changing of the observed scene due to factors such as waves, causing the camera to constantly produce image bursts.

As such, marine conservationists 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 [30, 84],

microbial sampling [41], and human-interaction response monitoring [66]. The use of drones for this type 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 [27, 79, 166, 170].

2.5.1 Utilising Photo-id Aides in Cetacean Conservation

Performing manual analysis of photo-id surveys with the goal of creating a catalogue can be extremely time consuming and labour intensive. As such, over the years there have been multiple photo-id aides developed. These have advanced at pace, gradually reducing the amount of human interaction and pre-processing of the data seen by these systems. A description of existing programs and academic literature is provided below, with an overview in Table 2.1.

Catalogue Management Systems

Due to the potential size of photo-id catalogues, especially in geographic locations with large resident populations, many aides focus on the management of these catalogues using databases. The first of these, FinScan, was developed in 2000 [98]. This is a semi-automated photo-id assistant whereby the user imports images taken during fieldwork. 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). The 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, as the FinScan algorithm is not rotation invariant. Whilst FinScan 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 [67]. 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 [87] 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.

CatRlog [109] is an R Shiny based catalogue management system which allows for photo-identification of individual cetaceans. Researchers enter descriptions manually for unique markings located on either the dorsal fin or fluke of the individual. One of catRlog's main advantages is the ability to be hosted locally on machines via RStudio whilst also being capable of handling catalogues hosted on cloud-based storage solutions. This allows for efficient id verification, as multiple experts can id images at the same time each uploading results to the central catalogue. If any discrepancies arise, such as an individual given a different id by multiple experts, this centralised approach allows for the catalogue manager to easily determine the final id and disseminate this to other catalogue users. CatRlog is also capable of automated print-friendly photo-id catalogue creation.

Non-Deep Learning Systems

Multiple systems which utilise non-deep learning computer vision are also available to aid cetacean researchers. Karnowski *et al.* propose using Robust PCA to subtract background from underwater images to help detect captive bottlenose dolphins and track their movements through multiple distinct areas [106]. CurvRank is an algorithm developed by Weideman *et al.* [230] 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 against 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. Whilst initially developed for cetaceans, CurvRank has also found use in land-based photo-id surveys due to its robustness. For example, Kulits *et al.* [118] utilise CurvRank and SEEK [18] as the basis for a human-in-the-loop system for African elephant (*Loxodonta africana*) photo-id.

Deep Learning Systems

In recent years the use of deep learning has been explored, inching forward towards a fully automatic photo-id system. One of the main labour costs in the creation of photo-id catalogues is the processing of survey data. This is usually cropping the images collected down to just

the fins in the image, removing unneeded background. Photo-ID Ninja[3] is designed to speed up this processing. Users 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, detected using an object detection model. Work is also ongoing to extend Photo-ID Ninja to allow for individual identification of the New Zealand common dolphin (*Delphinus* spp.) using pigmentation, utilised due to the low chance of other prominent markings becoming present on individuals [80]. 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 [34]. Current reported accuracies for this pigmentation matching are a top-1 accuracy of 90.6%, top-5 accuracy of 93.6%, and a mean average precision (mAP) of 80.8%.

Quiñonez *et al.* propose a CNN based system capable of cetacean object detection, detecting four distinct classes: `dolphin`, `dolphin_pod`, `open_sea`, and `seabirds` [169]. Morteo *et al.* [150] perform semi-automatic fin measuring using fin shape in order to aid in population monitoring. This technique, based on [234], does not require the user to trace 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.

Bouma *et al.* provide a system focusing on metric embedding learning to photo-id individual New Zealand common dolphins (*Delphinus spp*). This research utilises Photo-ID Ninja to detect and crop fins before they are passed for identification, focussing on common dolphins as data subjects. The system is capable of achieving a top-5 accuracy scores of around 93% [34], although it should be noted here that the data utilised to achieve this score is not currently publicly available. An attempt to obtain this data was undertaken, but no response was received. As such, it is not possible to determine how distinct each individual is in the dataset. Based on the figures presented in the paper, it seems no segmentation of the cropped fins is performed before they are embedded. As such, some noise in the embeddings will be present.

FinFindR is an algorithm developed by Thompson *et al.* [207] which allows for inputted images containing bottlenose dolphins to be identified. FinFindR works with uncropped images and automatically detects any dolphin present 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 [185], with clustering of individuals achieved using Ward's variance minimising clustering [229]. Reported accuracies for FinFindR currently stand at a top-1 accuracy of

[3]Photo-ID Ninja: photoid.ninja

88%, top-5 accuracy of 94%, and top-50 accuracy of 97%. It should also be noted that FinFindR has currently only been tested on bottlenose dolphins and work is still ongoing.

Work undertaken by Georgetown University and Google in the area of cetacean photo-id has also provided promising results [126, 143, 214]. 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.

Recent work undertaken by Lee *et al.* proposes a novel architecture for cetacean identification [125]. 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 [179]. The resultant output is then passed to a post processor which re-aligns and normalises the fin. The most significant features of the fin are then extracted and passed to a VGG based system [192] combined with a novel V2BC component for identification, although results achieved for identification in the paper are sparse.

Maglietta *et al.* propose DolFin [139], a Speeded-Up Robust Features (SURF) based identification system [17] built upon in work undertaken by Renò *et al.* [176] for identifying individual 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 [87]. DolFin also makes use of a "fin mask extraction" unit capable of detecting, segmenting, and post-processing fins before they are identified using SURF, although details on this unit are unclear. This system is truly fully autonomous, although due to the SURF based system would only be capable of working with Risso's dolphins. Other cetacean species do not develop as prominent scarring, which would greatly reduce the effectiveness of SURF based feature extraction systems.

Online Tracking Systems

Whilst the use of photo-id catalogues for resident populations is beneficial for researchers in the animal's local area, some cetacean species traverse large geographical distances, sometimes travelling between continents. In these cases it would not be feasible for one research group to record sightings. This is where online tracking systems play an important role. These websites started by allowing users to enter their own sighting data into an online database, allowing for tracking of individuals over large areas. In recent years however the focus has shifted to performing detection and identification online from unlabelled sighting images. This allows groups of users such as citizen scientists to contribute to the database, as no knowledge of the existing catalogue or training in photo-id is required.

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 [105], utilising elements of ArcFace [58] and DeepFace [201] 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 tail stocks however, and work is currently ongoing in this area [42].

FlukeBook[6] is a fully automatic photo-id system capable of identification of multiple cetacean species. 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 the conservation space [23]. FlukeBook makes use of both CurvRank and FinFindR when working with bottlenose and spotted dolphins (*Stenella frontalis*). Whilst FlukeBook is capable of detecting and identifying individual cetaceans, the fact that it is hosted online with all submissions freely searchable may be a disadvantage for some cetacean researchers. Local photo-id catalogues are closely guarded due to the effort and expense required to collect the data, and as such some groups may prefer a local automated photo-id solution over one which requires them to hand over their data to a third party.

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

Summary of Available Photo-id Aides

As can be seen there is a wide variety of photo-id aides currently available to researchers, each fulfilling a different need depending on the researcher and how comfortable they are with automating parts of the photo-id process. If they prefer a fully manual approach to analysis but require storage solutions, FinScan and FinBase provide excellent catalogue management. DARWIN can then extend this through automating existing catalogue databases.

Manual analysis is extremely time consuming however, leading to modern systems allowing for some automation of processing. For fast cropping Photo-ID Ninja can be utilised, greatly reducing down the sizes of images and removing unneeded background. Algorithms like CurvRank and FinFindR can then be utilised to aid in identification.

Whilst all of these systems perform their intended task well, they are mostly stand-alone. This would require researchers to make their own data pipelines should they wish to host locally, such as Bouma *et al.* and their use of Photo-ID Ninja for detection before embedding using deep learning to provide catalogue matching [34]. Should researchers feel comfortable with open-sourcing their catalogues and matching process, whilst also working with a supported cetacean species, then systems like HappyWhale or FlukeBook are most appropriate, requiring little data preprocessing whilst not requiring researchers to host multiple algorithms or solutions locally, reducing the need to create their own custom data pipelines.

It is the work undertaken by the aforementioned solutions which motivates this thesis. A fully automated photo-id system must be capable of the following:

Operating with no data preprocessing: Researchers should not need to process data from fieldwork cameras before it can be handled by the system.

Detection: The system must be capable of detecting dorsal fins present in input images. This allows researchers to pass the system all images taken in the field to reduce human workload and allow the system to operate on data which has not been pre-processed. The system must be capable of detecting multiple fins in the same image.

Segmentation: Once fins have been detected, it must be possible to segment them from the image. This aides the photo-id process by removing other fins, which may contain prominent markings, as well as background such as the sea, which may be feature heavy due to waves.

Photo-ID: The system must be capable of identifying individuals.

Handle unseen individuals: The system must be capable of handling individuals which have never been seen before, and are thus not in the existing catalogue.

Locally deployable: It must be possible to deploy the system on local hardware, allowing researchers to keep full ownership of their catalogue.

An overview of how current photo-id aides meet these requirements can be seen in Table 2.1. Only Maglietta *et al.* propose a fully automated photo-id aid which can be locally ran, requires no data preprocessing, and is capable of handling unseen individuals. However as previously stated this solution utilises SURF [17]. This algorithm, like its predecessor SIFT [137], excel at detecting well defined features in images. This is appropriate for Maglietta *et al.*'s data subjects, Risso's dolphins, due to the well defined scarring patterns present on their dorsal fins [145]. The algorithm would fail when used on other cetacean species however due to a lack of well defined features on the fin, where even prominent markings used for photo-id fail to be extracted. See Section 3.2.1 for an investigation into the use of feature extraction algorithms on bottlenose dolphin dorsal fins. Furthermore, both SURF and SIFT are patented algorithms. As such this may increase the cost of developing or using a system based around these algorithms, an open source solution should be utilised if possible.

Table 2.1 A comparison of available photo-id aides.

System	Data Preprocessing	Detection	Segmentation	Photo-ID	Handles Unseen Individuals	Locally Deployable
FinScan [98]	✓	✗	✗	✓	✓	✓
FinBase [67]	✓	✗	✗	✓	✓	✓
DARWIN [87]	✓	✗	✗	✓	✓	✓
catRlog [109]	✓	✗	✗	✓	✓	✓
CurvRank [230]	✓	✗	✗	✓	?	✓
Karnowski *et al.* [106]	✗	✓	✓	✗	–	✓
Photo-ID Ninja[3]	✗	✓	✗	✗	–	✗
Quiñonez *et al.* [169]	✗	✓	✗	✗	–	–
Morteo *et al.* [150]	✓	✗	✗	✓	✗	✓
Bouma *et al.* [34]	✓†	✗	✗	✓	✓	✓
FinFindR [207]	✗	✓	✗	✓	?	✓
Georgetown University & Google [126, 214]	✓	✗	✗	✓	✗	✗
Lee *et al* [125]	✗	✓	✓	✓‡	?	✓
DolFin [139]§	✗	✓	✓	✓	✓	✓
HappyWhale[4]	✗	✓	✗	✓	✓	✗
FlukeBook[6]	✗	✓	✗	✓	✓	✗

? It is unclear if the system is capable of handling unseen individuals based on the literature available.
† Utilises Photo-ID Ninja for detection.
‡ Sparse on detail regarding photo-id performance.
§ Utilises SURF for photo-id, thus unsuitable for some cetacean species.

2.6 Summary

This Chapter presents the key ideas required to understand and appreciate the novel work proposed in this thesis. An 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. This Chapter examines one such subcomponent utilised in the development of an automatic photo-id system, the cetacean detector. This component takes images captured during photo-id surveys and locates regions of interest (RoIs) - defined as areas in which a dorsal fin breaches the waterline. This Chapter will discuss the requirements a detector must meet, as well as its training and hyperparameter optimisation.

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 RoIs. This detector will only be required to identify one object class, dolphins. These detected regions can then be passed further down the system pipeline for photo-identification.

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 RoIs must be detected, and the technical requirements the system must perform under, this is actually a complex problem.

3.1.1 Environmental Requirements

Firstly the area in which this system is to be deployed, 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

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.

sea swell. Further to this, cetaceans 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 downstream, the system must still be able to separate them into individual detections to prevent misclassification.

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 (artificial) eye. Especially from a distance and in choppy waters, fins and waves often have extremely similar shape and structure. 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 2.4, CNNs see input images as a matrix of pixel values. When training an object detection system the CNN is 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 similar 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. When working in an open water environment

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 but keeps the identifying notch visible.

with live animals, there is no guarantee how the animal will approach the camera, and thus the detector must be generalisable enough to handle this.

Furthermore how the animals breach the water is also extremely variable. Breachings may occur in 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) with the animal travelling perpendicular as this provides the best chance capture markings - 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, although they may still contain identifiable markings. The 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 in photo-id surveys there are only two conditions that need to be worried about; swell and lighting. This is due to most research groups limiting travel in rough seas 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 [194]. 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 photo-id surveys take place during daylight hours. This can lead to large amounts of glare in images, especially on clear days. The detector should be invariant to these conditions.

3.1.2 Technical Requirements

On top of 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 [103]. Thanks to the pace of research in this area, 2020 saw the release of detection architectures which can perform operations in real-time such as EfficientDet [202] and YOLOv4 [29]. Current accuracies 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 them, 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 asleep or during the day whilst they are on surveys. 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 2.4, the output of detection systems can be provided in different formats. In bounding box detection systems the detected objects are described 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. Bounding box based detections are limited in their ability to remove background noise however, with only the background outside of the box removed.

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, allowing for the removal of as much background as possible. Both semantic and instance segmentation methods allow the detector to utilise pixel-wise mappings to remove background noise. Pixel wise labelling is far more labour intensive and costly to produce compared to bounding box labelling, and systems capable of pixel level detections are often slower than bounding box detectors. Utilising our requirements as defined in Section 3.1.1, specifically that the detector must be capable of reducing an overlapping pod to its individual component animals, the use of pixel wise mappings at an instance level would be preferable over semantic or bounding box level detections. 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 is discussed in more detail in Section 3.2.1.

Furthermore, any system performing cetacean detection from photo-id survey data must be capable of working with large scale images. In most image based tasks where deep learning is utilised images fed to the network are downscaled, typically to sizes such as 224x224 to allow for faster training and a reduction in overall network size. Downscaling images reduces the number of pixels in the image, which by definition reduces the amount of information present as pixels values need to be pooled (one pixel needs to now display what multiple would have previously). For most detection tasks this would not be an issue, and indeed if this project was solely a cetacean detection task there would be no issues with downscaling. This detector is not stand-alone however but rather the first stage of a pipeline of networks with the end goal of photo-identification. The identification task relies on potentially minute details in the fin such as notches; any downscaling of the image at the detection stage runs the risk of removing potentially identifiable information in the fin. As such, the image must only be reduced in size once it is certain that no identifying information will be lost. As this cannot be guaranteed at the stage of detection, the detector must be capable of operating on images without resizing.

3.2 Deciding on Approach

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, and ultimately the overall automatic photo-id system, would be the use of either bounding boxes or pixel-wise mappings. As mentioned previously, the use of pixel-wise mappings would allow for a greater removal of background

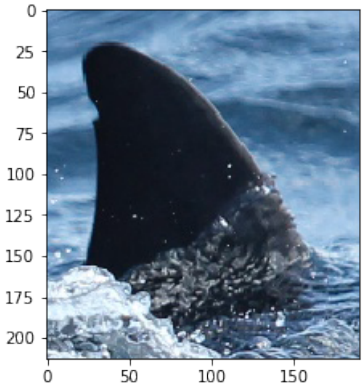

Figure 3.3 An example manual crop utilised in bounding box suitability testing.

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 at Newcastle University, 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 RoI, the dorsal fin, was centred and prominent representing an optimal output from a bounding box detector.

Feature Extraction with SURF

To begin, processing of the cropped images focussed on the use of feature extractors such as Speeded-Up Robust Features, also known as SURF [17]. Like its predecessor Scale Invariant Feature Transform (SIFT) [137], SURF is invariant to scale, a major advantage for use with cetacean survey data where the RoI's size may change depending on when the image of the dorsal fin breaching the water is captured. If SURF 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 SURF was performed over the entire cropped image. This proved unfruitful however, 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 SURF was allowed to explore.

Reduction of the search space available to SURF 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, SURF would only be performed in areas of the image where pixel values fell within this range. An example result of SURF 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. Issues arise however where areas of water are also within this thresholding. Because of this, colour thresholding before SURF 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 SURF is incapable of extracting relevant prominent markings from the species in the image, Indo-Pacific bottlenose dolphins (*Tursiops aduncus*). For example, in Figure 3.3, a notch is clearly present on the dorsal fin which is a good marker for individual identification. However, when performing SURF on this dorsal as seen in Figure 3.4, note how this notch has not been detected by SURF, which has instead detected an area above the notch which contains no identifiable information.

Feature extraction methods such as SURF are also incapable of extracting other identifiable markers such as fin shape. As such, the use of feature extractors was deemed improper for this use case. It is important to note here that the use of feature extractors may be appropriate for cetacean species other than this project's data subjects of bottlenose and white-beaked dolphins. For example, the use of both SURF and SIFT have been shown to be appropriate to aid in identification of individual Risso's dolphins [139, 176].

Background Removal with GrabCut

Testing the suitability of SURF as described in Section 3.2.1 highlighted the need for complete background removal before identification would be possible with bottlenose and white-beaked dolphins. In order for bounding boxes to be a viable option in this scenario, a

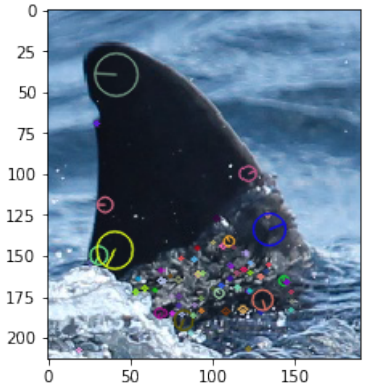

Figure 3.4 An example manual crop showing the result of SURF feature extraction when thresholded based on RGB colour values.

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 pixel labelled 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.

GrabCut, an algorithm proposed by Microsoft Research [181], 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, the algorithm 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 SURF testing. Again, issues arose when performing GrabCut on the cropped image data. The algorithm struggled to understand which parts of the image were background and foreground, resulting 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.

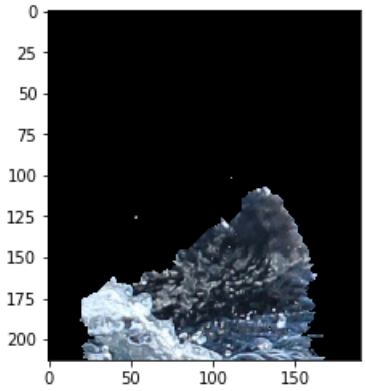

Figure 3.5 An example manual crop showing the result of GrabCut background removal.

3.2.2 Instance Segmentation Architectures

One of the major decisions that must now be made 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 application domain, 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 from more novel parts of the project (notably the identification of the individual animals). Further, as this project is introducing deep learning methods to an application domain where it is not commonplace, the project needs to be able to convince researchers in the 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 considered for this task; U-Net [179] and Mask R-CNN [93]. Both of these architectures work in different ways. Vuola *et al.* provide a more detailed comparison between the two models [222], 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 only one output is required. For example taking U-Net's original use case of biomedical imaging, the model is able to efficiently segment a group of cells into individual components through boundary estimation to locate the outer edges of the cells. This 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 R-CNN utilises a multi-stage architecture (described in more detail in Section 2.4.5) allowing the model to place each detection on its own binary output mask.

This is extremely important for the outlined use case. As the detector will be used as part of a larger system for automatic photo-id, any detections made will need to be passed to the identifier as a stand-alone image. If U-Net was utilised for the detection stage, whilst initially being more efficient than Mask R-CNN, further processing of the binary output mask would still be required to split this into it's individual components. In contrast, if Mask R-CNN 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 R-CNN for this stage.

Another factor which must be decided upon when starting development of a deep learning system is the language and framework to be used for development. With regards to language this was a fairly easy decision; a large proportion 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 [2] and Facebook's Torch (of which PyTorch is the most actively developed) [161], 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 the project at this stage. 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 R-CNN in Tensorflow for this project, Matterport's Mask R-CNN implementation [225] has been adapted.

3.3 Initial Testing of Mask R-CNN

In order to build a Mask R-CNN 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 R-CNN 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 [57] which contain a large corpus of varied classes, only CIFAR-100 [115] 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 [20], for reasons discussed in more detail in Section 2.5. Some images included in the iNaturalist dataset [217] 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 Newcastle University's Marine MEGAfauna Lab. The dataset was curated during a 2015 conservation effort undertaken in Zanzibar, Tanzania, to determine the status of Indo-Pacific bottlenose dolphins in the area [188]. 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 perform this conversion, the provided images must first be labelled. This was achieved using the VGG Image Annotator software, known as VIA [62]. Other labelling software such as LabelImg [212] 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 R-CNN. Second, the tutorial data provided by the Mask R-CNN 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 [154], 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 R-CNN 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 R-CNN.

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. 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 R-CNN 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 R-CNN 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.

3.3.2 Transfer Learning

Whilst the Zanzibar dataset provides experimental data similar to that which the Mask R-CNN 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

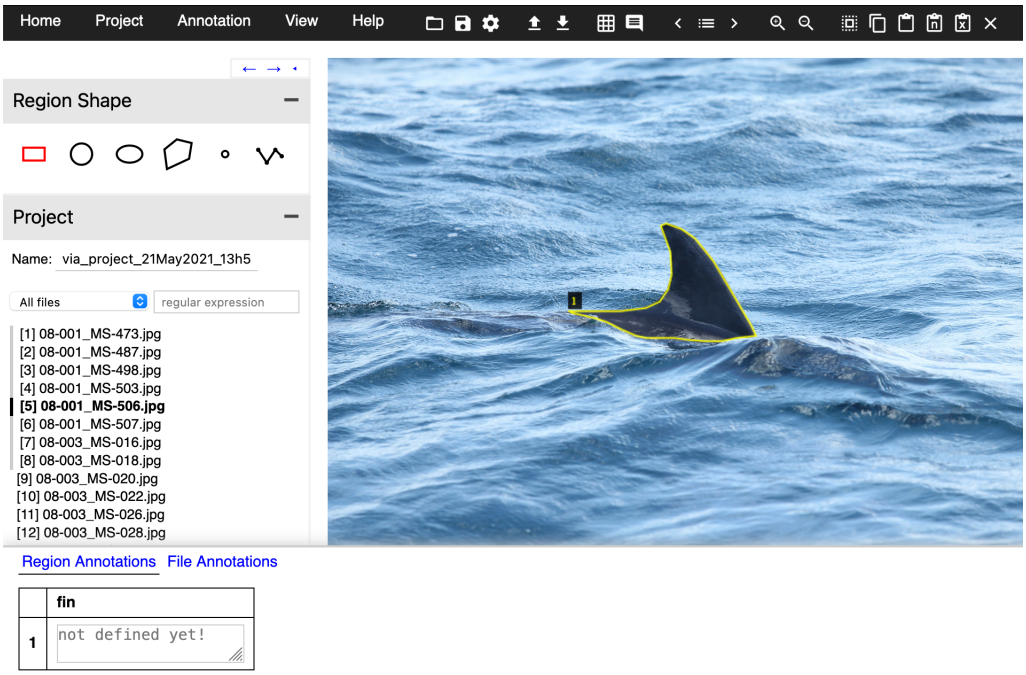

Figure 3.6 An example image showing the labelling processes using VIA.

and accurate models. As such, this dataset alone would not be enough to train the cetacean detector. One way to approach 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 approach available 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 [57] or Microsoft's Common Objects in Context, more commonly known as MSCOCO [129]. 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.* [160].

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 and 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 R-CNN

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 architecture is chosen. For the cetacean detector, it was decided that a ResNet50 backbone would be utilised. Matterport's Mask R-CNN implementation allows for the use of a ResNet50 or ResNet101 backbone, both standard variants of ResNet which are 50 and 101 layers deep respectively [94]. 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 although a significant increase in training time was observed.

Once a backbone architecture was chosen, it can then be 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. In the case of a model being trained from scratch without transfer learning, the weights of each layer are randomly initialised and then manipulated through backpropagation to achieve a desired model output.

In transfer learning however, the model's starting weights are not initialised randomly. Instead, the weights of the trained network hosted on the model zoo are used as a starting point. This replicates the final state of the model trained on the larger dataset.

As previously mentioned, there are multiple different models available in the zoo, all trained on a large variety of benchmark datasets. Each dataset produces a model with different final trained weights. Before applying transfer learning to the ResNet50 architecture chosen it is important to make an informed decision as to which benchmark dataset the model should be initialised from.

For this project it was decided that the ResNet50 weights trained on MSCOCO [129] 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 R-CNN has in recent years been well documented in literature for a variety of tasks [54, 71, 247] including in land-based photo-id system, with Kulits *et al*. utilising MSCOCO as a transfer learning dataset when training a modified Faster-RCNN system for African elephant re-identification [118]. As Mask R-CNN builds on Faster-RCNN, it was deemed reasonable to assume MSCOCO would also work well for pretraining a Mask R-CNN based system.

When utilising an MSCOCO pre-trained architecture for a Mask R-CNN 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 R-CNN models, such as those which deal with the per-pixel masks. This is because these layers require the same number of neurons as dataset classes, similar to a fully connected layer. If the MSCOCO weights were utilised at these final layers for the task at hand there would be a mismatch between the number of classes in MSCOCO, 80, and in the Zanzibar dataset, 1 (plus `background`).

Once the backbone architecture 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, only the model heads are selected. These are layers required for the Region Proposal Network, the pixel classification, and masking layers of the model. For the purposes of hyperparameter tuning, whether the model weights are randomly initialised or loaded from a pre-trained model can be selected.

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. Figure 3.7 shows examples of data augmentations found in literature.

When augmenting data it is extremely important to understand a dataset's problem domain to ensure that any transformations are realistic and expose the model trained 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.

Figure 3.7 Examples of data augmentations found in literature. Left: Randomly erasing parts of images, from Zhong *et al*. [254]. Right: Augmentations to simulate screenshot capture. Defocus blur (top), motion blur (bottom) from Atapour-Abarghouei *et al*. [13].

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. 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 many 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 R-CNN 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.

3.4 Mask R-CNN Model Selection

When training a Mask R-CNN model there are a large range of hyperparameters, or user defined values, which must be set before training can occur. These hyperparameters each have influence on the final model's performance, and 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 R-CNN. 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 the goal of producing the optimal overall model for the task of cetacean detection, both on the Zanzibar dataset and on other similar datasets.

3.4.1 Detection Hyperparameters

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 before notifying the user. This reduced both the false positive rate and duplicate detection rate of the model. An example of this can be seen in Figure 3.8. Note how in the left image more detections are present than in the right image, however the extra detections are either part of an existing detection (yellow) or do not contain enough fin to identify the individual (red).

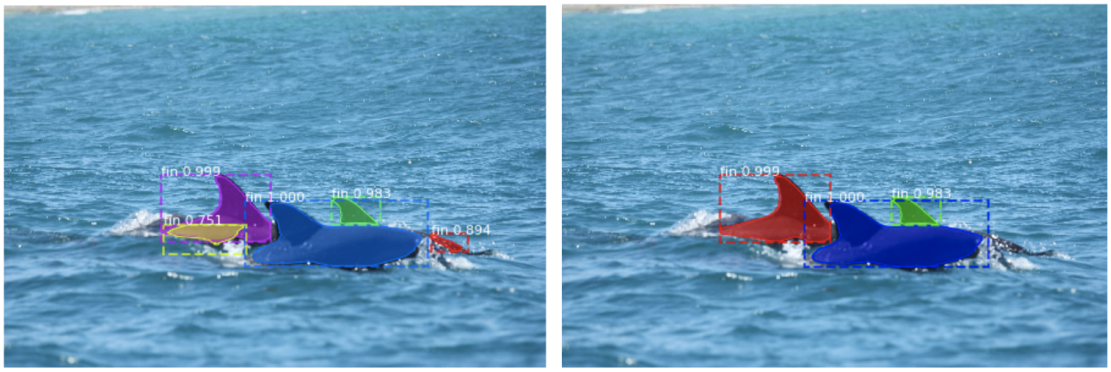

Figure 3.8 An example image showing the effect of minimum detection confidence thresholding in Mask R-CNN detections. Left: A threshold of 0.7. Right: A threshold of 0.9.

3.4.2 Training Hyperparameters

The vast majority of hyperparameters influence the training process. Selection of the optimal hyperparameters is however 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 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 infeasible due to time and resource constraints however, therefore a technique known as grid searching was 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 each combination of defined hyperparameter values. Once each model has been trained, they are then evaluated to determine the best hyperparameters. In some cases an acceptable model will be found

during the initial grid search, however this may not be the case. In this situation 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 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 adjusted using a scheduler. These allow for the learning rate to be modified 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 rate optimisers can also be used. These optimisers provide an alternative to SGD (an overview of which can be found in Section 2.3.1) 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 SGD, as discussed in Section 2.3.1. During hyperparameter tuning of the Mask R-CNN, two optimisers were chosen for evaluation.

The first, SGD with restarts (SGDR) [136] allows for decreases in the learning rate through a process known as cosine annealing, whereby the decrease follows a cosine waveform. This results in 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 [107], capable of achieving 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 [60] and RMSProp [209] 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.* [113].

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. As previously mentioned, due to how closely guarded cetacean photo-id catalogue data is and how expensive it is to collect, 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 [117]. 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 2.4.2, 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. The anchor scale provided to the RPN can be considered a hyperparameter as the best scales for the RPN to allow for the detection of objects regardless of their size must be determined.

3.4.3 Hyperparameter Tuning via Grid Search

Although only a few hyperparameters have been chosen to tune, 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 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 training run, can be seen in Table 3.1. This reduced hyperparameter tuning run still required significant amount of time and resources, running over three NC12 Microsoft Azure VMs, each with two Tesla K80 GPUs, taking approximately one week to complete the grid search producing a total of 50 models. Model runs were split between the VMs based on augmentation strategy with one VM running only *aug1*, the other *aug2*, and the final with no augmentation strategy. It should be noted here that this computational and time expense would most likely be reduced should the images used to train the Mask R-CNN not be so large, although the reasons for this decision are discussed in Section 3.1.2.

3.4.4 Model Selection Based on Grid Search

Once a grid search has been performed, the results can then be evaluated to determine if a suitable model had been found using the test set. All models trained were evaluated using MSCOCO's Mean Average Precision metric[1], a commonly used metric for segmentation tasks. This metric, commonly written as mAP@IOU[0.5:0.95], calculates precision-recall graphs for each dataset class at incremental IOU levels, from 0.5 to 0.95 in 0.05 steps. Once each class' precision-recall graph for a given IOU threshold has been calculated, the mean

[1] COCO mAP Definition: cocodataset.org/#detection-eval

Model Name	Weight Decay	RPN Anchor Scales	Optimiser	Augmentation Strategy	Pre-trained on MSCOCO?
20190829T1458	0.01	(16, 32, 64, 128, 256)	Adam	aug1	True
20190829T2020	0.01	(16, 32, 64, 128, 256)	Adam	aug2	True
20190830T0145	0.01	(16, 32, 64, 128, 256)	Adam	None	True
20190830T0714	0.01	(16, 32, 64, 128, 256)	Adam	aug1	False
20190830T1443	0.01	(16, 32, 64, 128, 256)	Adam	aug2	False
20190830T2019	0.01	(16, 32, 64, 128, 256)	Adam	None	False
20190902T0946	0.01	(16, 32, 64, 128, 256)	SGDR	aug1	True
20190904T2004	0.01	(16, 32, 64, 128, 256)	SGDR	None	True
20190905T1813	0.001	(32, 64, 128, 256, 512)	SGDR	aug1	True
20190905T1826	0.01	(16, 32, 64, 128, 256)	SGDR	aug2	True
20190905T2202	0.001	(32, 64, 128, 256, 512)	Adam	None	True
20190905T2336	0.001	(32, 64, 128, 256, 512)	Adam	aug1	True
20190906T0332	0.001	(32, 64, 128, 256, 512)	SGDR	None	True
20190906T0851	0.01	(32, 64, 128, 256, 512)	Adam	None	True
20190907T0932	0.001	(16, 32, 64, 128, 256)	Adam	aug1	True
20190907T0933	0.0001	(32, 64, 128, 256, 512)	Adam	aug2	False
20190907T0934	0.01	(32, 64, 128, 256, 512)	SGDR	None	False
20190907T1451	0.001	(16, 32, 64, 128, 256)	Adam	None	True
20190907T1500	0.0001	(32, 64, 128, 256, 512)	Adam	aug1	False
20190907T1545	0.01	(32, 64, 128, 256, 512)	Adam	aug2	False
20190907T2026	0.0001	(32, 64, 128, 256, 512)	SGDR	None	True
20190907T2126	0.001	(16, 32, 64, 128, 256)	Adam	aug1	False
20190907T2215	0.001	(16, 32, 64, 128, 256)	SGDR	aug2	True
20190908T0202	0.0001	(16, 32, 64, 128, 256)	Adam	None	True
20190908T0352	0.01	(32, 64, 128, 256, 512)	Adam	aug2	True
20190908T0417	0.0001	(32, 64, 128, 256, 512)	Adam	aug1	True
20190908T0957	0.0001	(32, 64, 128, 256, 512)	Adam	None	False
20190908T1102	0.0001	(32, 64, 128, 256, 512)	Adam	aug2	False
20190908T1204	0.001	(16, 32, 64, 128, 256)	SGDR	aug1	True
20190908T1939	0.001	(16, 32, 64, 128, 256)	Adam	aug2	False
20190908T2043	0.0001	(32, 64, 128, 256, 512)	SGDR	aug1	True
20190908T2139	0.0001	(16, 32, 64, 128, 256)	Adam	None	False
20190909T0723	0.0001	(16, 32, 64, 128, 256)	Adam	aug1	False
20190911T1922	0.01	(16, 32, 64, 128, 256)	Adam	aug2	False
20190912T0045	0.0001	(16, 32, 64, 128, 256)	Adam	aug2	False
20190912T0608	0.0001	(16, 32, 64, 128, 256)	SGDR	aug1	True
20191101T1633	0.0001	(32, 64, 128, 256, 512)	SGDR	aug2	True
20191101T2104	0.001	(8, 16, 32, 64, 128)	SGDR	aug2	True
20191102T0140	0.01	(32, 64, 128, 256, 512)	SGDR	aug1	True
20191102T0615	0.01	(8, 16, 32, 64, 128)	SGDR	aug2	True
20191102T1051	0.0001	(16, 32, 64, 128, 256)	SGDR	None	False
20191102T1528	0.001	(32, 64, 128, 256, 512)	SGDR	aug2	True
20191102T2006	0.0001	(8, 16, 32, 64, 128)	SGDR	aug2	True
20191103T0044	0.0001	(8, 16, 32, 64, 128)	SGDR	aug1	True
20191103T0520	0.001	(8, 16, 32, 64, 128)	SGDR	None	True
20191103T0959	0.01	(32, 64, 128, 256, 512)	SGDR	aug2	True
20191103T1441	0.0001	(8, 16, 32, 64, 128)	SGDR	None	True
20191103T1921	0.0001	(16, 32, 64, 128, 256)	SGDR	aug2	True
20191104T0011	0.001	(8, 16, 32, 64, 128)	SGDR	aug1	True
20191104T0450	0.01	(8, 16, 32, 64, 128)	SGDR	aug1	True

Table 3.1 Hyperparameter values used for each grid search run when training the Mask R-CNN model on the Zanzibar data.

of these values is derived giving an overall mean average precision score for all classes at a given IOU threshold; these thresholds are explained in more detail in Section 2.4.3.

By evaluating over multiple thresholds the models can then be compared and their performance more easily understood and ranked, as well as allow for the determination of an acceptable loss in IOU overlap. For example if all models were evaluated using mAP@IOU[0.5] only, it may be the case that all models achieve a similar high score, making

it difficult to determine which model will be best for the task. However if too high a threshold is used, for example mAPIOU[0.95], it is unlikely that any model will achieve a high score as this would require constant near pixel perfect detection.

Figure 3.9 shows a visualisation of the mAP@IOU[0.5:0.95] scores for all models trained in the grid search, the raw scores can be seen in Appendix A. At mAP@IOU[0.5] there is a large gap in model performance with model 20190830T1443 having the lowest mAP@IOU[0.5] of 0.73 and model 20190905T1813 having the highest at 0.94. This shows that the combination of hyperparameter values provided to the model before training have a significant effect on the model's overall performance, although even the lowest score here is still high.

At mAP@IOU[0.75], whereby detections would overlap with 75% of pixels in the ground truth mask, the minimum model performance has dropped significantly with model 20190830T071 achieving a score of 0.49. The highest score at this threshold is model 20191102T0140 with a score of 0.81; this model achieved an map@IOU[0.5] score of 0.92, only two percentage points behind the best model at that threshold. This again shows the need for hyperparameter tuning when selecting models, as they are shown here to have a significant effect on how well the models perform at higher thresholds.

This effect is even greater when comparing map@IOU[0.85] scores, with the worst performing model, 20190830T0714, achieving a score of just 0.17 whilst the best model, 20190902T0946, achieves a score of 0.50, a difference of 33%. Model performance drops significantly at the highest threshold with four models achieving an mAP@IOU[0.95] score of 0.016, with most models achieving a score of 0.0. This is to be expected however as it would be highly unlikely that any model, regardless of hyperparameters, would be able to perform near perfect pixel level detections on the test set data.

Whilst Figure 3.9 provides some indication of overall model training, using it to determine the most appropriate model for the task at hand is difficult given the number of models trained. To achieve this, the list of models was reduced to only those which achieved the best mAP@IOU[0.5, 0.75, 0.85] scores. The thresholds 0.5 and 0.75 were chosen to remin consistent with other segmentation literature [31, 208, 228]. The 0.85 threshold was chosen as some models trained still achieve impressive results here, allowing for more model filtering. Further, the pixel-wise detections of fins is required to filter as much background noise as possible and so finding high performing models at top thresholds is important.

When filtering, the top five performing models at each threshold were extracted and then combined into one list. If a model achieved top five ranking at multiple thresholds it was only included in the list once, resulting in a list of ten best performing models. The

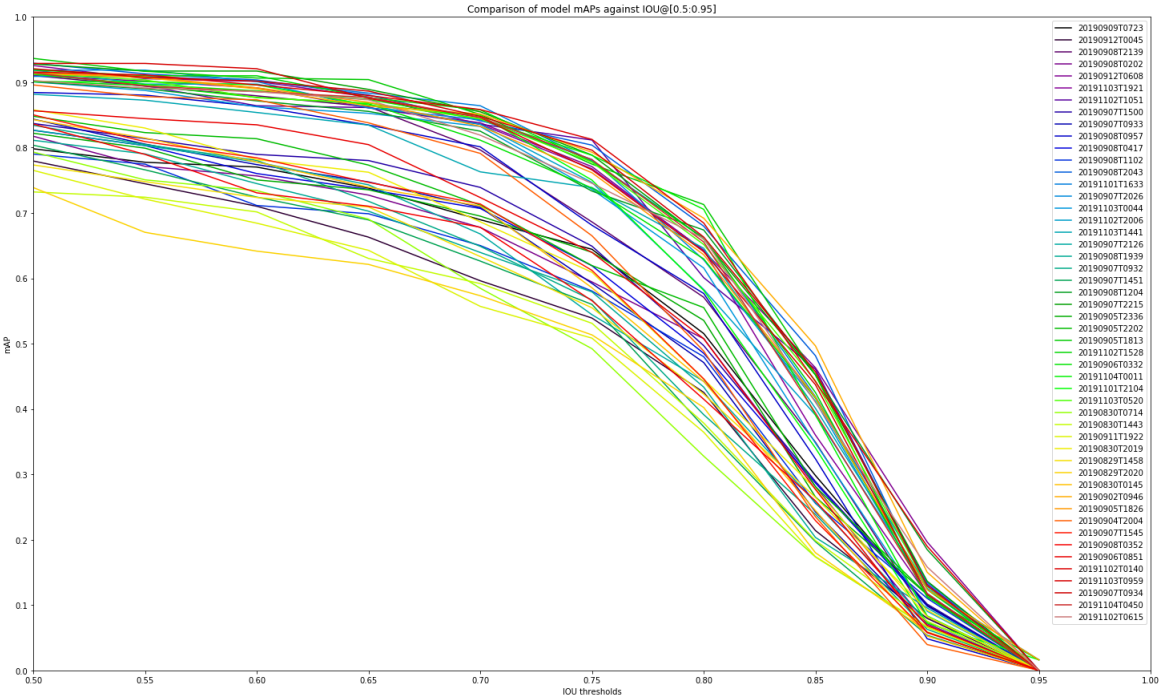

Figure 3.9 mAP@IOU[0.5:0.95] scores for all models trained in the Mask R-CNN Zanzibar dataset grid search. See Table 3.1 for each model's hyperparameters.

mAP@IOU[0.5, 0.75, 0.85] scores for these best performing models can be seen in Figure 3.10.

When deciding on which model hyperparameters are best for the task of cetacean segmentation, it is important to find a model with a high mAP@IOU[0.85] score. As the model will be used to perform segmentation before fine-grained classification, it is important the model is capable of removing as much background from the input image as possible. Any background included in the segmentation may adversely effect the photo-id process. Using this as criteria, model 20190902T0946 was selected as the best performing model. The model achieves an mAP@IOU[0.85] score of 0.5, an excellent result given the difficulty of the segmentation task. The model also performs well at the other evaluation thresholds, achieving mAP@IOU[0.5, 0.75] scores of 0.91 and 0.79 respectively. These scores verify the model is capable of segmenting cetacean fins from background with as little noise being included in the segmentation mask as possible.

An interesting point to note here is that 20190902T0946 did not achieve the highest mAP@IOU[0.5, 0.75] scores. As previously mentioned, these thresholds are often the ones included in segmentation literature to evaluate model performance. If just these thresholds were used for model selection, 20190902T0946 would not have been chosen. This highlights

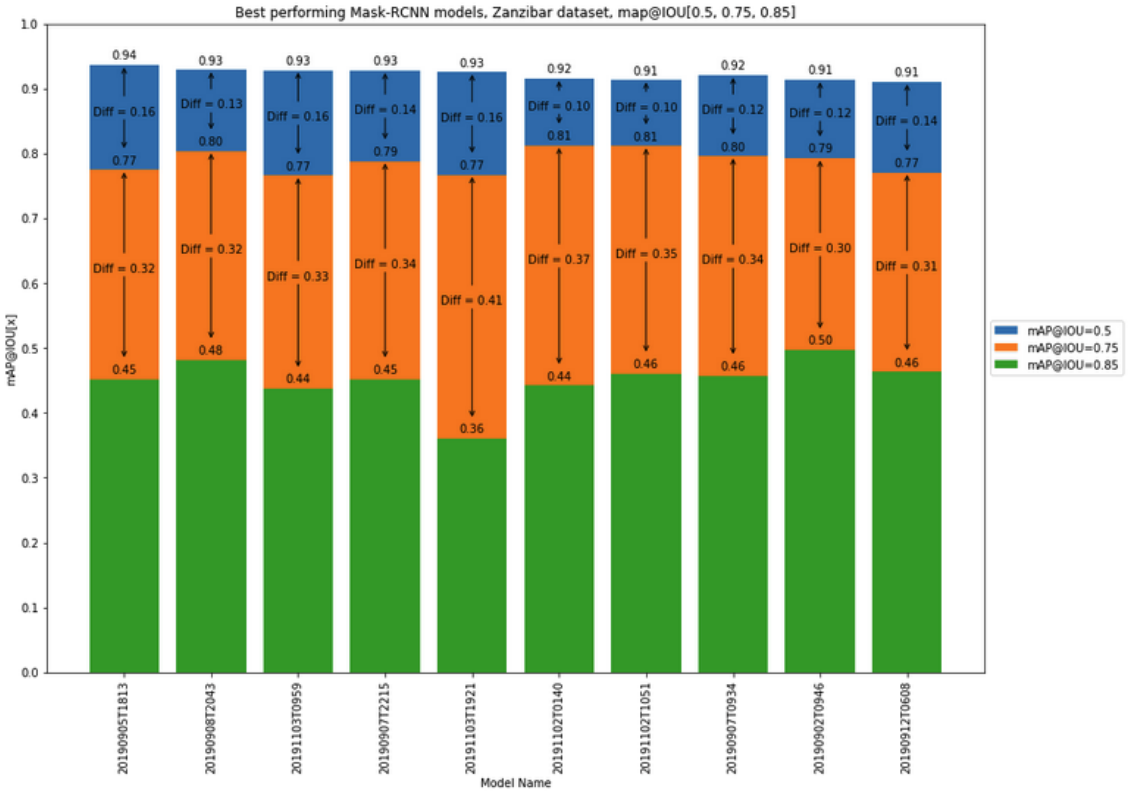

Figure 3.10 mAP@IOU[0.5, 0.75, 0.85] scores for the best performing Mask R-CNN models trained on the Zanzibar dataset. See Table 3.1 for each model's hyperparameters.

the need to select models based on metrics which make sense for the task at hand. As the model is required to remove as much background noise as possible, using a high threshold for evaluation makes sense. Thresholds higher than 0.85 were not utilised due to the low performance of all models at this threshold, although 20190902T0946 also achieves one of the best mAP@IOU[0.9] score of 0.150. Only one model, 20191102T0615 achieves a better score at this threshold, 0.158, however this model achieves lower performance at the chosen evaluation thresholds of 0.5, 0.75, and 0.85.

3.4.5 An Evaluation of Optimal Model Hyperparameters

As discussed in Section 3.4.4 a filtering of the trained Mask R-CNN models was performed to determine the best model hyperparameters for the task of cetacean instance segmentation, with model 20190902T0946 ultimately being selected for future use. This model's hyperparameters, along with those of the other nine best performing models, can be seen in Table 3.2.

Model Name	Weight Decay	RPN Anchor Scales	Optimiser	Augmentation Strategy	Pre-trained on MSCOCO?
20190902T0946	0.01	(16, 32, 64, 128, 256)	SGDR	aug1	True
20190905T1813	0.001	(32, 64, 128, 256, 512)	SGDR	aug1	True
20190907T0934	0.01	(32, 64, 128, 256, 512)	SGDR	None	True
20190907T2215	0.001	(16, 32, 64, 128, 256)	SGDR	aug2	True
20190908T2043	0.0001	(32, 64, 128, 256, 512)	SGDR	aug1	True
20190912T0608	0.0001	(16, 32, 64, 128, 256)	SGDR	aug1	True
20191102T0140	0.01	(32, 64, 128, 256, 512)	SGDR	aug1	True
20191102T1051	0.0001	(16, 32, 64, 128, 256)	SGDR	None	True
20191103T0959	0.01	(32, 64, 128, 256, 512)	SGDR	aug2	True
20191103T1921	0.0001	(16, 32, 64, 128, 256)	SGDR	aug2	True

Table 3.2 Hyperparameters of the best performing Mask R-CNN models on the Zanzibar dataset. Subset of Table 3.1.

The hyperparameters of the best performing models provide an interesting insight into the training process. All ten of the models were trained using SGDR. This is interesting, as the current trend in deep learning network training is to utilise Adam [107]. Furthermore each model trained utilised transfer learning, with each model's parameters being initialised from a trained MSCOCO model provided by the model zoo. This highlights the need to utilise pre-trained models, especially cases where relativity small amounts of data are available to train a model from scratch.

Half of the best models utilise the *aug1* data augmentation strategy, defined in Section 3.3.4. The smallest RPN Anchor Scale, (8, 16, 32, 64, 128), has not been utilised by any of the best models, and the value of the weight decay hyperparameter is split between the three possible values. These splits highlight the need for a robust and in-depth hyperparameter search, as with the majority of hyperparameters searched no clear trend can be identified.

3.4.6 Limitations of the Model

As with any neural network trained on real world data, the best performing model, 20190902T0946, is not perfect. As can be seen through the mAP scores in Figure 3.10, the model still fails to correctly detect in some instances. This section examines under what conditions 20190902T0946 fails for the purposes of model transparency.

Sometimes there are instances where the detection fails to capture all of the individual. This seems to occur when parts of the animal are poorly lit or void of unique markings leading to a consistent matt dark colour scheme. An example of this behaviour can be seen in Figure 3.11's green detection. Here, the model has correctly detected the part of the animal's body which is above the waterline, but has failed to fully detect the matt dark dorsal fin. This poor detection may also be influenced by the undetected individual close behind the detected animal.

Figure 3.11 also shows another good example of environmental conditions which cause the model to fail to correctly detect an individual. Due to wave movement covering a part of the left animal's body, the model has split this individual into two separate detections.

Figure 3.11 Left: The input image passed to the cetacean detector. Right: The detection masks produced by the model overlaid onto the image. Note how the cetacean on the left has been split over two masks due to occlusion from a small wave.

The model also struggles in cases where the image contains an animal under the waterline but due to the clarity of the water is partly visible in the image. In this case the model often detects the individual under the waterline. Due to these individuals not being useful for identification purposes however, they were not labelled in the dataset and thus are deemed to be misclassifications when evaluating the model. Figure 3.12 shows an example of this issue occurring.

Figure 3.12 Left: The image passed to the cetacean detector. Right: The detection masks produced by the model overlaid onto the image. Note how the blue detection is of an individual under the waterline and only partly visible, and thus useless for identification purposes.

The Zanzibar dataset contains a large number of images which contain other vessels as well as dolphins. This is due to the large marine eco-tourism industry within Zanzibar [24, 189]. Whilst this issue may not be present in other survey areas where this model may be deployed, it still denotes an example of the model failing. In this case often parts of the boat,

or a combination of the boat and the humans on the boat, may cause the model to incorrectly identify a grouping of pixels as a dolphin. An example of this can be seen in Figure 3.13.

Figure 3.13 Left: The image passed to the cetacean detector. Right: The detection masks produced by the model overlaid onto the image. Note how the red detection is a misclassification. The model believes a section of the boat's hull and the leg of the human to be a dolphin.

All of these mis-detections have an impact of the overall evaluation score of the model. These mis-detections will then be passed further down the system pipeline to be classified as individuals. To reduce the chances of this happening, a robust post-processing technique was developed.

3.5 Mask Post-Processing Techniques

At this stage, the system is capable of detecting cetaceans at an individual pixel level. Before these detections can be passed to the identification module, some post-processing of the output must be performed to allow for both a reduction in the computational expense of operating on the detector's output as well as ensuring that no potentially important information which will assist in an identification is lost.

3.5.1 Handling Multiple Detections

When an image is ran through the Mask R-CNN detector, the number of outputs can vary depending on how many detected objects have a confidence score higher than 90% (as discussed in Section 3.4.1). If no detections reach this threshold the image is discarded from further processing. This can occur if, for example, an image is taken by accident capturing the vessel's floor. If a single detection reaches the threshold, the resultant mask is passed downstream.

Thanks to the tendency for cetaceans to travel in pods however, it is unlikely an image will contain only a single `dolphin` detection. In these situations the Mask R-CNN will

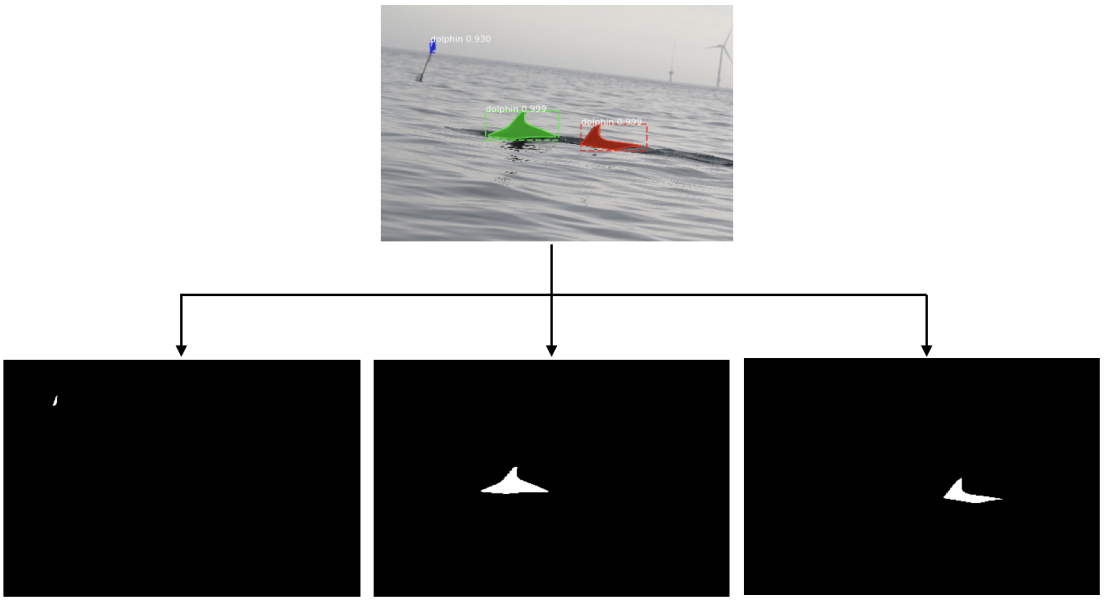

Figure 3.14 A visualisation of the three `dolphin` detections for an input image produced by the Mask R-CNN detector. The detections are highlighted by the blue, green, and red overlays on the input image, top. The resultant detection masks once split are shown bottom, where a `dolphin` object is displayed in white.

output multiple detection masks, one per object. To handle this, the first stage of the post-processing methodology is to separate multiple detections for a single image. This ensures each `dolphin` detection is handled independently, mitigating the potential for an identification to be influenced by the others. An example of this behaviour can be seen in Figure 3.14. Here, an image inputted to the Mask R-CNN detector has produced three detections which are above the threshold. As such, they have been split into three output masks for individual processing. Note that the first detected masks, visualised with a blue overlay on the input image, has been incorrectly detected as `dolphin` with a high confidence resulting in a binary mask. Further post-processing must be capable of handling background which has been incorrectly labelled, removing it before individual identification occurs.

3.5.2 Morphological Transformations

In some situations a detected mask may contain an area of background inside the detection. This can be thought of as a hole in the detection, as seen in Figure 3.15 (Left). Using *a priori* knowledge of cetaceans, which would rarely if ever be captured with a hole, it can be deduced that a hole in the detection is highly unlikely and may cause a loss of useful identifying information. As such, any holes which are present in the masks must be filled.

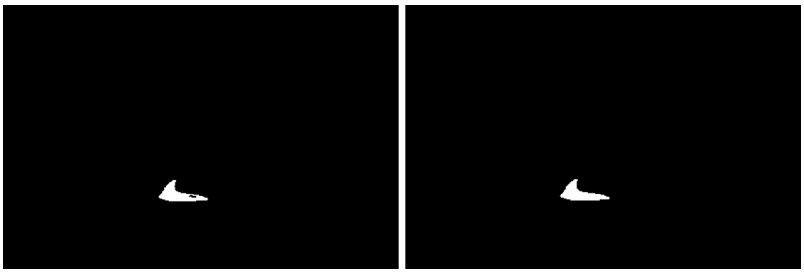

Figure 3.15 Left: A detection mask before closing has been applied. The detected `dolphin` object is displayed in white. Note the cluster of black background pixels inside. Right: The same detection mask after closing. Note the pixels which make up the hole have been converted to `dolphin`.

This is achieved using morphological transformations, a set of operations which allow for the automated manipulation of the internal structure of binary images such as masks. The two fundamental morphological transformations are *erosion*, which erodes away the boundaries of the masked object, and *dilation*, which increases the size of the object by pushing the boundary out into the background space. These two operations can be utilised in various combinations to perform more complex transformations.

In order to remove a cluster of background pixels inside of a detection, each mask is *closed* - dilated then eroded. This has the effect of removing any holes present inside the mask, as can be seen in Figure 3.15 (Right). If no holes exist, the operation is still performed however the mask remains unchanged. By performing closing, the system ensures that no potentially identifiable information is lost as a result of an incomplete detection.

3.5.3 Background Subtraction

Now that the masks have been cleaned using morphological transformations, it is possible to utilise them to perform background subtraction. This is an extremely important step in producing an accurate individual classification based on the detected `dolphin` object by ensuring a minimal amount of background noise is passed to the identification system.

As both the input image and resultant mask can be represented as matrices, these can be manipulated utilising a *bitwise and* operation such that if $\text{pixel}_{i,j}$ in the input image is denoted as background in the mask, the values of $\text{pixel}_{i,j}$ can be set to [255, 255, 255] (white). This has the effect of whiting out any pixels not detected as part of the `dolphin` in the input, whilst keeping the pixels detected as `dolphin` intact.

An example of background subtraction utilising cleaned masks can be seen in Figure 3.16. Using the same input image as in Figure 3.14, it can be seen that the *bitwise and* operation

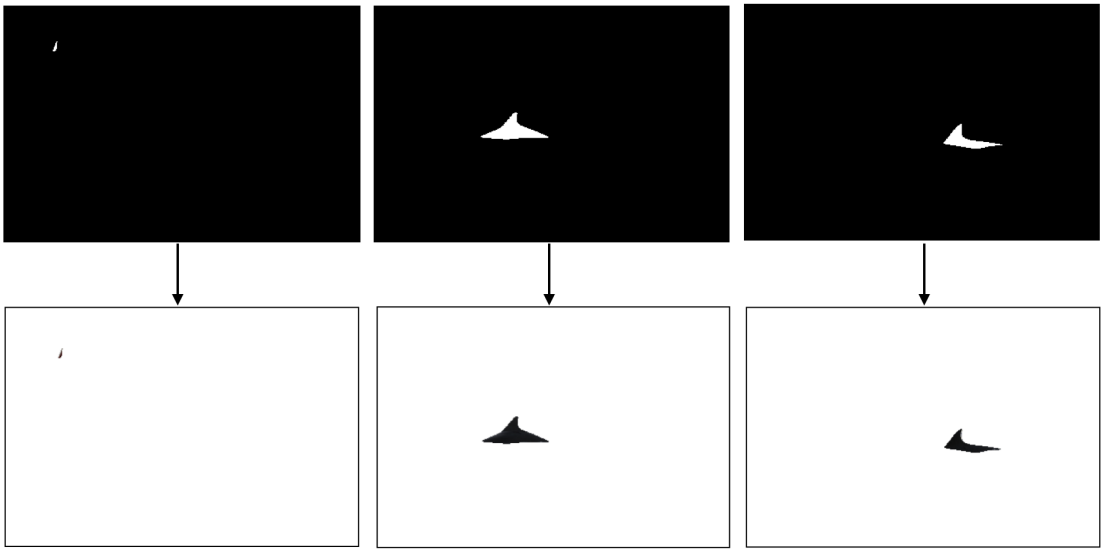

Figure 3.16 A visualisation of the three `dolphin` detections from Figure 3.14 before and after background subtraction.A border has been added to the background subtracted images for clarity.

whites all pixels in the image except those which have been classified as `dolphin` for each of the three output masks. Note at this point that the erroneous classification still remains.

Whilst the background subtraction aims to reduce noise passed downstream to the individual identification module, it will not be possible to remove all noise. It may be the case, such as in Figure 3.17, that some background pixels have been mislabelled as `dolphin` and are connected to at the edges of the object. As a result, morphological transformation and background subtraction are unable to remove the mislabelled pixels. This may affect the accuracy of identification downstream unless the system is robust enough to deal with this.

3.5.4 Colour Thresholding Mask Components

In some cases a single detection mask may consist of multiple components. This may occur if, for example, an area of splash has been erroneously included as part of a detection. As cetaceans cannot be made up of multiple disjoint components, it is known that some of these must be noise and can be removed.

The outer layer of a cetacean's skin is often a consistent grey colouring. This information can be utilised to filter out noisy components of the mask during post-processing. By comparing the colour composition of each detected object against a calculated *dolphin-like* threshold, it is possible to discard mask components which have been erroneously detected.

In order to be able to compare each mask component's composition against a *dolphin-like* colour threshold, the values of the threshold must first be determined. To achieve this, each

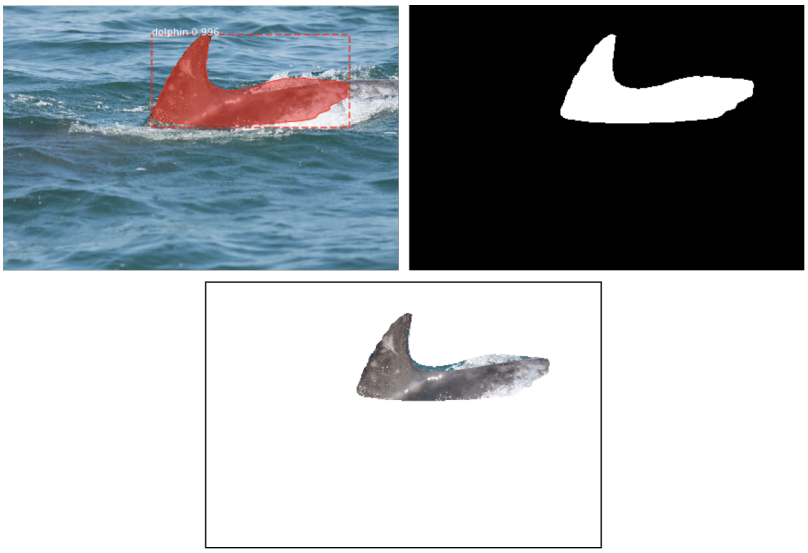

Figure 3.17 Top Left: A visualisation of the `dolphin` detection for an input image produced by the Mask R-CNN detector, alongside the confidence score. The detector has incorrectly labelled some pixels as `dolphin`. Top Right: The resultant detection mask after morphological transformation. Bottom: The resultant output image after *bitwise and* operations performed between the input image and the cleaned detection mask. Note the mislabelled pixels are present after cleaning and background subtraction. A border has been added for clarity.

image in the Zanzibar dataset was ran through the Mask R-CNN detector. Histograms of the three RGB colour channel pixel intensities for each object classification (`dolphin` or background) were recorded, giving a total of six histograms per image.

Once complete the histogram groups were combined to give six global pixel intensity distributions, which can be seen in Figure 3.18. From the charts it can be seen that, regardless of colour channel, there is a near inversion in the distribution of pixel intensities between those detected as `dolphin` and those not, strongly suggesting it is possible to determine if a component is erroneous based on its colour composition.

Using the global distribution histograms, a *dolphin-like* threshold was determined. For all masks detected as `dolphin`, 90% of the RGB pixel intensities are below [148, 148, 159]. The colour representing this threshold can be seen in Figure 3.19. In contrast only 50.19%, 38.24%, and 36.36% of the background pixels for each RGB channel respectively are below this threshold. As noise components in the mask are often areas of water or splash, these components will be likely much lighter in composition than cetaceans, and thus can be removed from the mask with confidence. An example of colour thresholding removing noise from a mask can be seen in Figure 3.20.

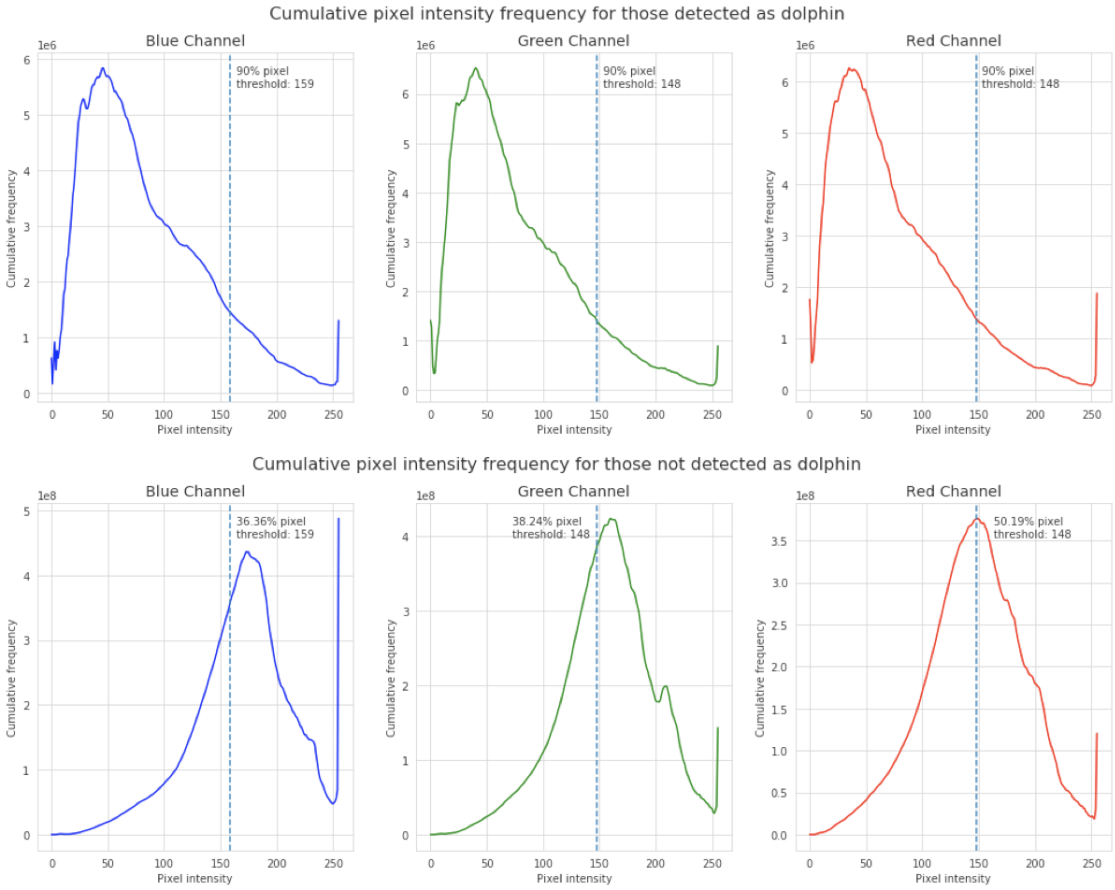

Figure 3.18 The global range of pixel intensities for each RGB colour channel, split by pixel classification.

Figure 3.19 A representation of the RGB threshold for *dolphin-like*, colour [148, 148, 159]. In the Zanzibar dataset, 90% of pixels detected as `dolphin` have intensities below this threshold.

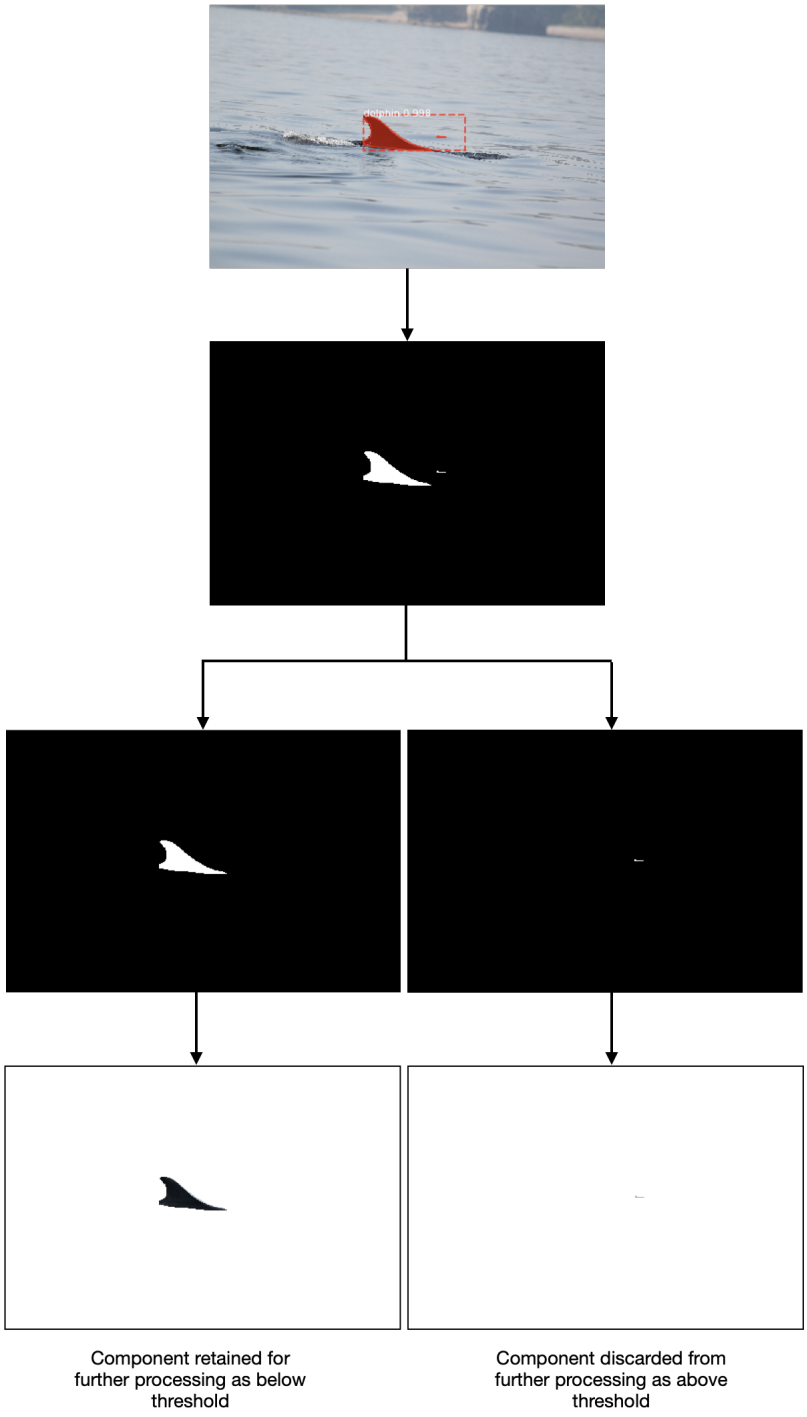

Figure 3.20 Workflow detailing colour thresholding to remove an area of disjoint splash which has been detected as part of a `dolphin` object. The detection mask is split into each component. The resultant background subtracted images are then colour thresholded. As a result, the erroneous splash is discarded.

During testing however it was found that when checking detections at an individual image level rather than globally, considering a mask component to be *dolphin-like* if 90% of the pixels were below the colour threshold was too restrictive. Utilising such a high percentage bar sometimes rejected valid detections which may have been over-exposed due to lighting conditions. As such, whilst the colour threshold was kept the same, it was found that reducing the percentage check to 50% provided enough leeway such that over-exposed but valid detections were kept whilst still rejecting a large portion of erroneous ones. Figure 3.21 shows an example detection which would be discarded if the 90% global threshold was used per image, but is retained if this is dropped to a 50% check.

It may be the case that multiple components in an image meet the conditions set by the threshold to be kept. In this case, each component of the mask is split into its own image, in a similar process as outlined in Section 3.5.1. If a mask only contains one component, then colour thresholding is not applied. This ensures no detections by the Mask R-CNN are completely ignored due to post-processing, preventing the discarding of a `dolphin` object mask which contains no disjoint components however is above the threshold, such as in the event of over-exposure. This condition also has the effect however of allowing fully erroneous detections to pass downstream, such as the flag detected in Figures 3.14 and 3.17. Any erroneous detections which pass downstream at this stage must now be handled by the identification module.

3.5.5 Cropping

At this point outputs from the Mask R-CNN detector have been post-processed to remove as much noise from the detected masks as possible, and these have been utilised to perform background subtraction. This results in an output image containing mostly white pixels surrounding a detected `dolphin` object. This image is the same size as the one inputted to the detector, which can often be many thousands of pixels, although the vast majority of these are now un-needed.

As a result, the images outputted from the background detector are now cropped down to contain just the object of interest. This has the effect of vastly reducing the image file size, which in turn reduces the computational expense of operating on them downstream. In Figure 3.22 for example, the input image to the detector is of size 3456x5184 pixels. After post-processing, the resulting output image is now of size 776x1350px, approximately a 94.2% reduction in image size whilst still keeping identifying information, such as the white pigmentation on the dorsal fin, present. This cropping also has the effect of centring the detected `dolphin` objects in the final output image.

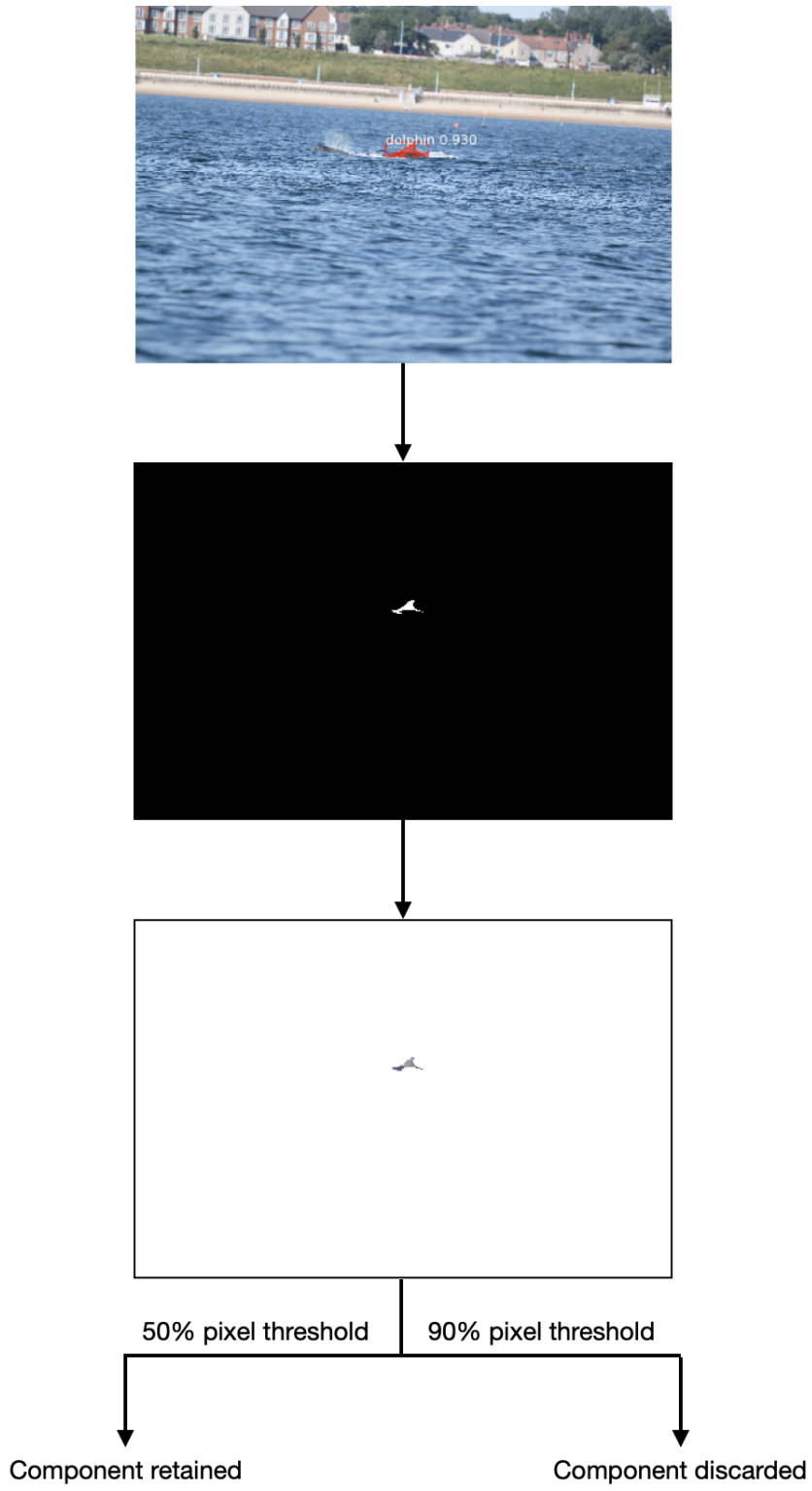

Figure 3.21 Workflow showing how utilising the 90% global threshold may lead to over-exposed detections being erroneously discarded. A `dolphin` object is detected whose mask consists of multiple components. One of the components, which contains a valid area of detection, is shown in stage 3 of the workflow. Checking to see if 50% of the pixels in the component are below the threshold retains the detection, whilst checking at 90% discards it.

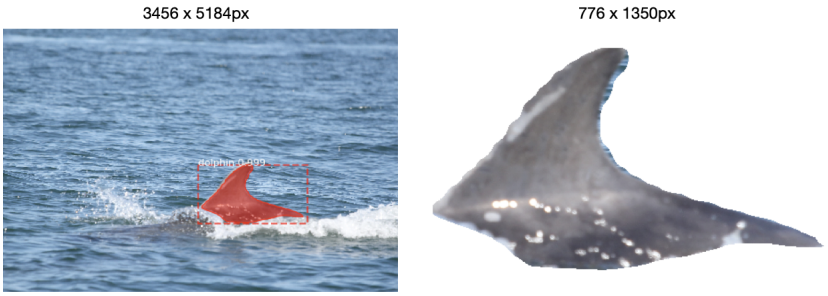

Figure 3.22 Left: An input image and overlaid detection mask. Right: The corresponding cropped output image after post-processing. Original image sizes are displayed above each image. Images have been resized for clarity.

3.6 Summary

This Chapter discusses the project's need for a model capable of cetacean detection, both from a technical and environmental perspective. The key reasons behind the use of instance segmentation masks rather than the relatively less computational and time expensive bounding boxes is explained, with evidence showing how the difficulty of the task influenced this move. Once the system requirements and underlying model architecture have been identified, the Chapter then examines the use of model hyperparameter optimisation to train a model capable of cetacean detection via instance segmentation masks. The use of model pretraining is also explored, and highlights the benefits of this approach even when using a pretraining dataset who's domain and distribution are vastly different to the final model's goal. Finally, the Chapter examines the use of post-processing techniques to improve the output of the Mask R-CNN's detections for use in downstream tasks.

The final result of this Chapter is a Mask R-CNN model capable of high mAP even at large IoU thresholds. The model's resulting detections are coupled with a post-processing pipeline capable of greatly reducing the amount, and improving the quality of, data which subcomponents further into the pipeline are required to operate on. This allows for greater computational efficiency downstream as well as ultimately more accurate and confident individual identifications.

It is of great importance that the trained Mask R-CNN model is capable of detecting cetaceans in photo-id data which has been gathered in different geospatial and temporal areas. It is also important that the detector is capable of similarly high levels of accuracy without re-training on data from that geographic area. Use of the detector on data collected in a different geospatial and temporal area, as well as the transformation of this data into a useable photo-identification dataset, is explored in Chapter 4.

Chapter 4

North Sea Data Collection

At this stage, the automatic photo-identification system in development is capable of detecting cetaceans in large panoramic images and post-processing these detections into a form ready for individual identification, as outlined in Chapter 3. So far, the detector has been trained and tested on data collected by Newcastle University's Marine MEGAfauna Lab whilst researching the abundance of Indo-Pacific bottlenose dolphins (*Tursiops aduncus*) off the coast of Zanzibar, Tanzania [188].

Whilst the Marine MEGAfauna Lab research heavily in the Indian Ocean around Zanzibar [16, 203–205, 231, 244], this is not the only place they operate [206, 245, 246]. In recent years, their work has begun to include more local waters such as the North Sea off the coast of Northumberland, UK [216, 243]. These waters are known to host a wide variety of marine mammals, with the Marine MEGAfauna Lab focussing efforts specifically on the bottlenose dolphin (*Tursiops truncatus*) and white-beaked dolphin (*Lagenorhynchus albirostris*) populations.

Near the end of the Mask R-CNN detection model's completion, the Marine MEGAfauna Lab were preparing to begin a large scale bottlenose and white-beaked abundance estimate and health assessment survey. As such the opportunity arose to validate the detector's generalisability on data which is similar in composition and purpose, however was collected in a different geographic location, at a different time, and containing different species to the data used to train and test the detector.

As a result, this Chapter discusses the collection of abundance estimate data in Northumberland, UK for the purpose of model and technique evaluation as outlined in Chapter 3. In order to achieve this evaluation, the photo-identification data was curated and transformed from a biological catalogue into a computer vision dataset known as *The Northumberland Dolphin Dataset 2020* (NDD20) - the creation of which will be discussed. The post-processing

of NDD20 and additional external data sources to produce a dataset capable of training an individual identification model is also examined.

4.1 Data Collection

The following Section provides context for the data collection survey, beginning by briefly outlining the geographic area in which the data was collected and discussion of why the area was chosen. Next the survey effort is discussed in detail, including a run-down of the methodology used, for the purposes of reproducibility.

4.1.1 The Survey Area

Data collection was conducted in and around the Coquet to St. Mary's Marine Conservation Zone (MCZ), located off the coast of Northumberland, UK. The MCZ, established in January 2016 through powers granted by the Marine and Coastal Access Act 2009 [1], covers approximately 40km of coastline from Almouth in the north to Whitley Bay in the south, extending outwards 7.5km at its greatest to cover an area around 192km^2. A map of the survey area and MCZ can be seen in Figure 4.1.

The area is of high importance, supporting a wide variety of marine life thanks to sections of intertidal and sub-tidal rock and sediment, making it fertile feeding grounds for the bottlenose and white-beaked dolphins which make use of the area. As a result of this fertility, as well as waters up to 30m deep in some places, the MCZ sees high levels of fishing activity - typically for crustaceans using pots [197]. Whilst these fishing vessels operate from a number of small ports throughout the North East of England, the MCZ itself lies close to the large Port of Blyth. As a result, the MCZ boundary provides a 250m buffer zone around the limits of the port in order to reduce economic damage. This survey region was selected as no previous surveying had been undertaken in the area for the purposes of cetacean abundance estimates and health assessment.

4.1.2 Survey Effort

Dedicated bottlenose and white-beaked dolphin photo-identification surveys were conducted in the MCZ between 19/07/2019 and 10/10/2019, with a total of 27 surveys undertaken. These were performed using a 5.6m rigid inflatable boat (RIB) with a 50 horsepower four-stroke outboard engine. All surveys began from Newcastle University's Blyth Marine Station, located in the Port of Blyth, before entering the MCZ.

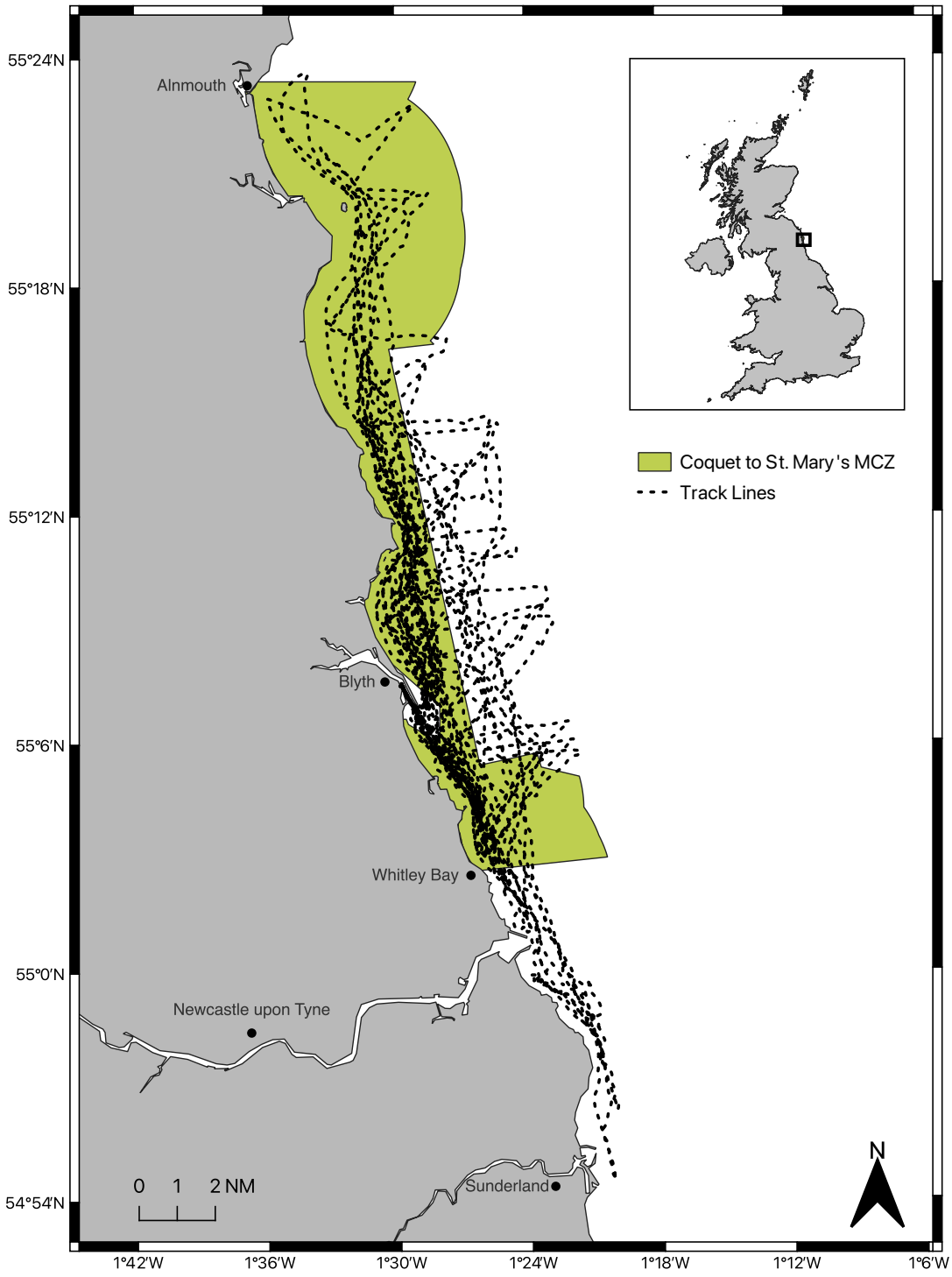

Figure 4.1 Map of the survey area, with the Coquet to St. Mary's MCZ highlighted. Track lines for all survey days are overlaid.

Surveys initially began by following set transect lines, traversing between the northern and southern-most points of the MCZ. Thanks to limited success encountering individuals strictly following transect lines however, the survey switched to more opportunistic surveying based on reports from two citizen science groups: the Newbiggin-by-the-Sea Dolphin Watch[1] and the North East Cetacean Project[2]. The use of citizen scientists for photo-id surveys has seen increased prevalence in recent years, with multiple studies producing promising results if access to groups of dedicated citizen scientists is available, like in Northumberland [8, 9, 11, 55]. Track lines showing movement of the vessel were recorded via GPS tracking, and can be seen in Figure 4.1. When dolphins were encountered, the time stamp was recorded alongside other effort data such as direction of travel, sea state, species, group size, and demographic composition.

Surveys were only conducted in Beaufort Sea States ≤ 3 [194] without heavy rain. Outside of these conditions surveying can become unsafe and the photographs unusable for photo-id because of swell and lens splash. Due to the nature of the North Sea, conditions outside of these restrictions can be common. Surveying was performed using the constant scanning method [142], with cues including sight of dorsal fins breaching the waterline, splashing, and leaping. For each survey the vessel was manned by at least two dedicated observers and a skipper, in line with other photo-id surveys [26, 188, 191].

Individual dolphins in an encounter were photographed randomly using a Canon EOS 550D Digital SLR with a Canon 70–200mm zoom lens, aiming to capture photographic data for every individual present. Camera settings can be found in Appendix B. Multiple photographs were captured of each cetacean over the course of the encounter to ensure identifiable information could be fully captured. When capturing an encounter care was taken not to approach individual cetaceans at an angle less than 30°, keeping as parallel as possible and to speeds no greater than 6 knots in order to prevent the cetaceans becoming stressed or injured as per Marine Management Organisation guidelines. All members of the survey team were trained in minimising wildlife disturbance through the WiSe Scheme by the Yorkshire Wildlife Trust[3], with the survey itself having the approval of Newcastle University's Ethics Board.

4.1.3 Field Season Summary

Of the 27 days where surveys were conducted 14 contained encounters. Of these, 12 were made up of bottlenose dolphin; only two were made up of white-beaked dolphin. No

[1] Newbiggin-by-the-Sea Dolphin Watch: facebook.com/groups/NEWILDDOLPHINMONITORINGPROJECT
[2] North East Cetacean Project: facebook.com/groups/NorthEastCetaceanProject
[3] WiSe Scheme: wisescheme.org

encounters contained both species. Groups were defined using the 10m chain rule [193]. Group size averaged 12 for bottlenose dolphins, typical for the species [186]. Altogether 40 individuals were identified and catalogued, broken down into 30 bottlenose and 10 white-beaked dolphins. Of all animals encountered, 27% were calves. They have been excluded from this analysis as they could not be considered independent due to reliance on their mothers, and had not yet developed permanent markings.

Images collected were processed for use in the photo-identification catalogue as to remove any images with no value, such as those which were out of focus or did not contain any cetacean. Animals present in the images were coded according to their distinctiveness as per the guidelines presented in [215]. Those coded D1 were considered very distinctive with little chance of misidentification, whilst those coded D2 were considered moderately distinctive with small prominent markings which could allow for a high chance of correct classification provided the image is clear. CF coded individuals were those which contained little to no identifying information which have a high chance of misclassification. Once coded, animals considered D1 and D2 were individually identified.

4.2 The Northumberland Dolphin Dataset 2020

The fieldwork season and data processing undertaken resulted in a photo-identification catalogue of bottlenose and white-beaked dolphins currently inhabiting the Coquet to St. Mary's MCZ. Photo-identification catalogues utilised in marine biology however are not in the form required for training or validating a computer vision model. As such, further processing was required to transform the catalogue into a dataset capable of both validating the instance segmentation model developed in Chapter 3, as well as training a model for fine-grained individual identification. This Section discusses the creation of the Northumberland Dolphin Dataset 2020 (NDD20) [211], the computer vision dataset created from the photo-identification catalogue.

4.2.1 Above Water Data

During fieldwork 4940 images were collected which contained part of a cetacean above the water line. Of these, 2201 images were considered usable for the creation of NDD20. Issues rending images unusable included a significant amount of water splash obscuring the cetacean, poor lighting conditions, or where individuals in a pod were too close together to accurately determine by eye the outline of all individuals.

From manual analysis it was determined that not all images contained enough identifying information for individual classification labels. Because of this, the decision was made to include multiple levels of granularity to the dataset. As all images contained part of a cetacean, each one could be labelled to allow for instance segmentation training. To enable this, each mask located was given the label `dolphin`. Next, masks could be provided a fine-grained species classification. Thanks to the difference in colour between bottlenose and white-beaked dolphins, every mask labelled for instance segmentation could also be provided a species label - either `BND` or `WBD` representing bottlenose and white-beaked dolphin respectively.

At the highest level of granularity, some masks contained enough information to allow for individual identification. If an ID could be attained with high confidence, likely from images with D1 or D2 coded individuals, the mask containing the individual was provided an `ID` label. Recent work has shown that publicly available datasets containing animals may aid poachers [19]. In response, to protect ongoing cetacean research efforts a pseudo-anonomisation has been performed. This however does not diminish the value of the dataset to computer vision researchers. It is not the case that images with sequential filenames were captured sequentially, and all individual IDs have been randomly allocated a numerical value rather than the code given to them by the Marine MEGAfauna Lab. All EXIF data found in the images has also been removed. The photo-identification catalogue itself is not publicly hosted at this time.

Data was labelled at a pixel level using the VGG Image Annotator [62] in a similar fashion to the Zanzibar dataset as discussed in Section 3.3.1. In order to speed up the process data labellers were employed through Newcastle University's Jobs On Campus service[4] to manually annotate the masks and label them for instance segmentation. Once complete all masks were checked for error correcting and consistency purposes. Afterwards, the extra labels for both species and individual level identification were added. As this required expert knowledge, data labellers were not utilised for this. Example above water images from NDD20 and the labels assigned can be seen in Figure 4.2.

4.2.2 Below Water Data

In addition to the above water imagery captured during fieldwork, NDD20 also contains below water images. Underwater photo-id is not as widely used compared to its above-water counterpart at present, however uses have been noted for certain species and environments in recent years [216, 223]. Whilst this data has not been utilised in the work on automatic photo-

[4]Newcastle University Jobs On Campus: ncl.ac.uk/careers/jobs/opportunities-on-campus/jobsoc/

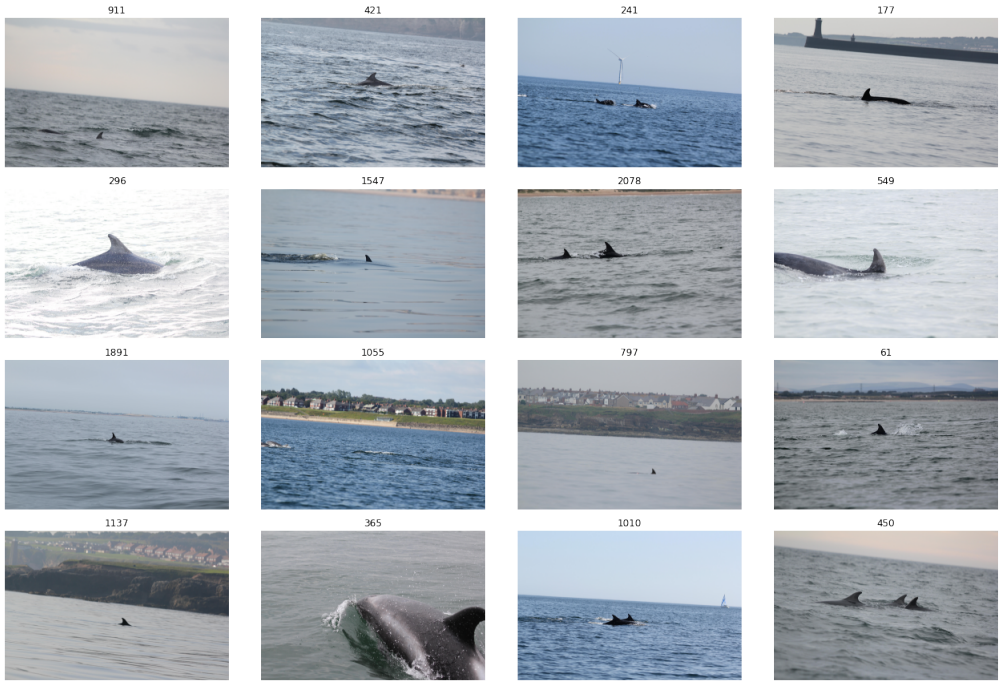

Figure 4.2 Example above water images from NDD20 with filenames displayed. Class labels for masks in each image are noted in Appendix C.

id presented in this thesis it is important to discuss all aspects of the dataset created. Images contained in the below water section of NDD20 are a subset of a much larger collection of images produced by the Marine MEGAfauna Lab during work in the Farnes Deep MCZ, a glacial trench situated approximately 11km from the Northumberland coast. Opportunistic surveys undertaken since 2011 have shown the area to contain a high abundance of white-beaked dolphin activity [216]. Data from these surveys takes the form of screen grabs from high definition video footage captured by a diver using GoPro Hero 3 and Go Pro Hero 4 cameras.

To mirror the above water section, there are 2201 below water images included in NDD20 labelled for multiple levels of granularity. As before, the first attribute level is `dolphin` to allow for instance segmentation. Unlike the above water images, all below water images contain at least one mask with an `ID` label. It is not the case that masks in the above and below image sets contain the same individual animal even if they have the same `ID` class label - the numbering systems are independent of one another. No species label is provided as all images are of white-beaked dolphins. Below water images are also labelled with an `out of focus` flag, denoting if the individual is deemed to be out of focus. Example below water images from NDD20 can be seen in Figure 4.3.

Figure 4.3 Example below water images from NDD20 with filenames displayed. Class labels for masks in each image are noted in Appendix D.

4.2.3 NDD20 Summary

As NDD20 is split into two related but distinct sets of images, a summary is provided below for each. Due to the nature of cetacean group dynamics, multiple images contain more than one individual animal. As a result, there are 2900 masks present in the above water set of 2201 images. These masks contain both a `dolphin` label for instance segmentation as well as either a `BND` or `WBD` label to facilitate species level fine-grained classification. It should be noted however that the distribution of species class labels is imbalanced, with 73% of masks being labelled `BND`. Some above water masks also contain an individual level `ID` label to allow for extreme fine-grained classification. Due to the nature of the task only 14% of masks contain an `ID` class label, with 44 distinct individuals present. Once again these classes are imbalanced presenting both a fine-grained and few-shot learning problem. The above water `ID` class label distribution can be seen in Figure 4.4. Many of the challenges associated with manual above water photo-id apply here too, particularly the likelihood that unique features are specific to one side of an animal's body which may not have been captured in the image.

Like its above water counterpart, the below water set also contains 2201 images each with at least one mask containing a `dolphin` label. Masks in the below water set are significantly larger than in the above water set, as far more of the cetacean is visible when captured below

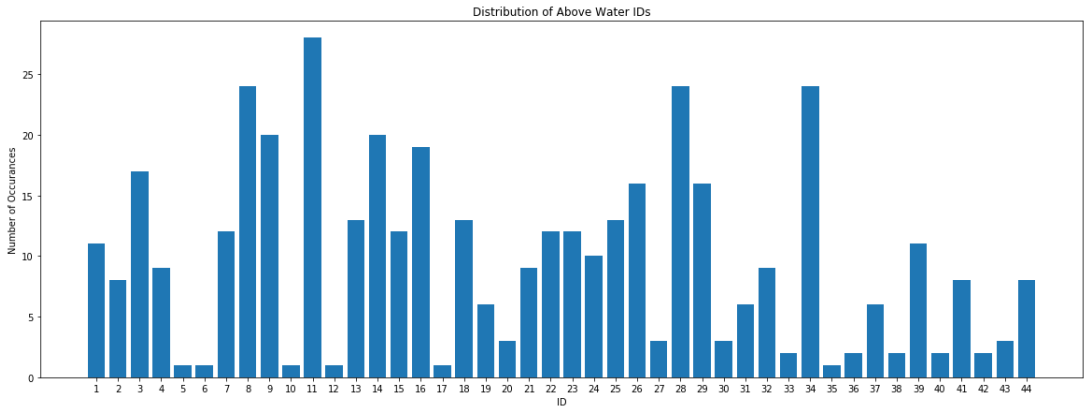

Figure 4.4 The `ID` class label distribution for the above water set of NDD20.

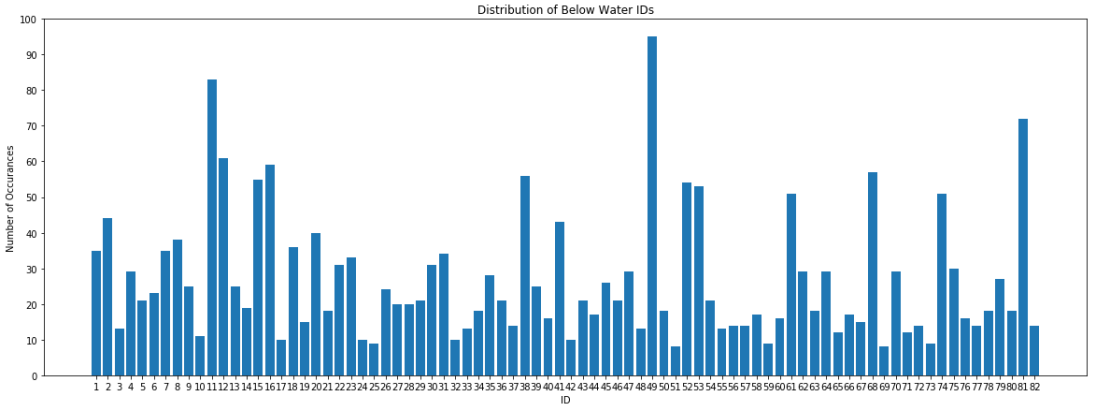

Figure 4.5 The `ID` class label distribution for the below water set of NDD20.

the waterline. Unlike the above water set however, all below water images also contain at least one mask with an `ID` class label, with 82 classes represented. The distribution of `ID` class labels in the below water set can be seen in Figure 4.5. This set represents a significantly more challenging fine-grained and few-show learning problem, both due to the higher number of classes as well as decreased image quality thanks to the nature of underwater photography. The main challenges are water clarity, affected by factors such as algae bloom and sunlight refraction which may both obscure areas of the individual useful for identification or add artificial markings which may hinder this.

NDD20 is small compared to traditional benchmarking computer vision datasets such as ImageNet [57] or those which are domain specific such as iWildcam [20] or Caltech-UCSD Birds 200 [233]. Even when compared to other related non-benchmark marine conservation datasets such as The Fishnet Open Images dataset [108] or SEAMAPD21 [33], the number

Table 4.1 A comparison of computer vision datasets capable of training models for individual animal identification, sorted by number of individuals.

Dataset	Species	Number of Images	Number of Individuals
Beluga ID 2022†	Beluga whale (*Delphinapterus leucas*)	5902	788
Leopard ID 2022‡	African leopard (*Panthera pardus*)	6795	430
Hyena ID 2022§	Spotted hyena (*Crocuta crocuta*)	3104	256
Cows2021 [75]	Holstein-Friesian cow (*Bos taurus taurus*)	10,402	186
BearID [51]	Brown bear (*Ursus arctos*)	4675	132
Multi Camera Pig Tracking [190]	Domestic pig (*Sus scrofa domesticus*)	380	33
Jaguar ID [210]	Jaguar (*Panthera onca*)	176	16
NDD20 (Below Water)	White-beaked dolphin (*Lagenorhynchus albirostris*)	2201	82
NDD20 (Above Water)	Bottlenose dolphin (*Tursiops truncatus*) & white-beaked dolphin (*Lagenorhynchus albirostris*)	2201	44

† Beluga ID 2022: lila.science/datasets/beluga-id-2022
‡ Leopard ID 2022: lila.science/datasets/leopard-id-2022
§ Hyena ID 2022: lila.science/datasets/hyena-id-2022

of images in NDD20 is much lower. Whilst it may be tempting to solely compare NDD20 to the above datasets however, it is important to note the difference in use-case. NDD20 is not designed for use as a large scale dataset for model pre-training like those considered benchmark. In contrast to the non-benchmark conservation datasets mentioned, NDD20 contains class labels which allow for more fine-grained classification. As such, it is better to compare the quality of NDD20 with other individual animal identification datasets.

Unlike other datasets for individual animal identification, NDD20 is unusual in that it covers multiple species. Other conservation datasets which cover a range of species do not include individual identification labels [20, 111, 217]. This combination of multiple species and individual class labels provides novelty to NDD20.

As seen in Table 4.1, whilst NDD20 is not the smallest dataset available for individual animal identification it is still on the lower end both in terms of number of images and individuals. This is due to the nature of the populations surveyed. The Coquet to St. Mary's and Farnes Deep MCZs are small geographic areas, resulting in smaller population catalogues. The size of the dataset has not limited scientific value however. NDD20 was accepted for presentation at the Fine Grained Visual Categorization workshop at CVPR 2020 [211]. To coincide with this presentation, the dataset was made public[5]. Furthermore, NDD20 was utilised as one of multiple datasets for use in a Kaggle competition hosted by HappyWhale [6]. This highlights the usefulness of NDD20 irrespective of size limitations.

[5]NDD20 download: doi.org/10.25405/data.ncl.c.4982342
[6]Kaggle competition: kaggle.com/c/happy-whale-and-dolphin

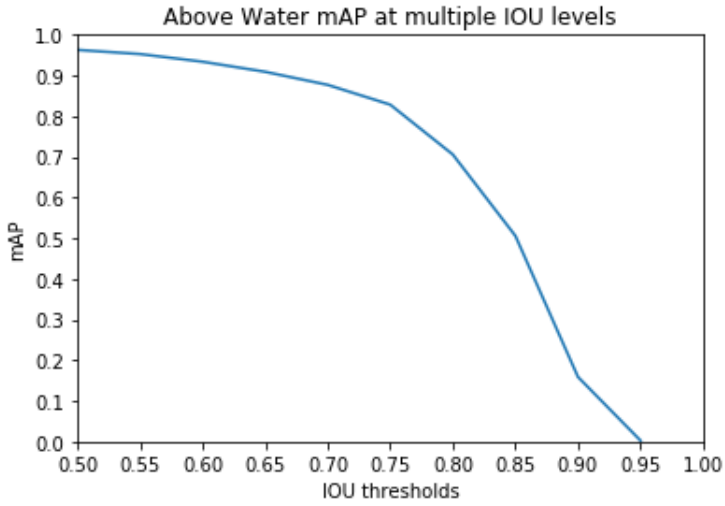

Figure 4.6 mAP@IOU[0.5:0.95] scores for NDD20 using model 20190902T0946 trained on the Zanzibar data.

4.3 Evaluation Using NDD20

Once NDD20 had been created it could then be utilised to evaluate the effect of changes in geography, time, and species on the Mask R-CNN based fin detector, as outlined in this Section.

4.3.1 Evaluating the Effect of Geography, Time, and Species Change

As discussed in Chapter 3, a Mask R-CNN model capable of above water cetacean detection was trained on indo-pacific bottlenose dolphin data collected in Zanzibar, Tanzania. One important requirement of the detector created is that it must be robust enough to output detection masks with high mean average precision (mAP) when operating on data from a different geographic or temporal area without re-training. The creation of NDD20 provides a valuable opportunity to test this requirement. Not only was the data collected in a different location and time, but both species of data subject (bottlenose and white-beaked dolphin) are not present in the Zanzibar dataset.

To test this requirement the best performing model found on the Zanzibar data in Section 3.4.4, 20190902T0946, was utilised to generate instance segmentation mask predictions for the above water set of NDD20. These model outputs were then evaluated against the labelled ground truth data to produce an mAP score over multiple IOU thresholds. The results of this evaluation can be seen in Figure 4.6.

Model 20190902T0946 still achieves a high mAP at multiple IOU thresholds without the need for re-training or fine-tuning on NDD20. Utilising the same evaluation thresholds as during model selection in Section 3.4.4, the model achieves mAP@IOU[0.5, 0.75] = [0.96, 0.83] on NDD20. This is in comparison to mAP@IOU[0.5, 0.75] = [0.91, 0.79] on the Zanzibar data. Interestingly, the model achieves a higher mAP at these IOU thresholds on NDD20. This is hypothesised to be due to the lack of other large objects in NDD20 in comparison to the Zanzibar data. For example, some images in the Zanzibar dataset contain other vessels as well as humans as a result of the data being captured in an area with high levels of eco-tourism [49]. This is not the case for data contained within NDD20. Whilst eco-tourism is present in and around the Coquet to St. Mary's MCZ, the levels are significantly lower than in Zanzibar. The evaluated model has been seen to struggle when presented with images containing tourist activity, shown in Figure 3.13. As NDD20 lacks this, the model's false positive rate may be reduced. Regardless, this evaluation presents evidence that model 20190902T0946 is robust enough to deal with data from a different geographic area, time, and cetacean species without the need for re-training or fine-tuning. This suggests the model could be deployed to aid in the speed up of future photo-id fieldwork seasons undertaken by conservationists.

4.3.2 Below Water Detection Baseline

Work was also undertaken to produce a baseline instance segmentation score for the below water set of NDD20. Here, a second Mask R-CNN model was trained using only the below water data divided into an 80:20 train-test split, with the `out of focus` flag ignored. This model achieved mAP@IOU[0.5, 0.75] = [0.97, 0.91], indicating a Mask R-CNN model is capable of accurate instance segmentation regardless of larger variation in the shape of masks in the below water data as a result of the underwater cameras being able to capture the individual cetaceans in a wider range of movement, as well as there being less differentiation between the background and the individual due to water conditions. The full range of mAP@IOU values can be seen in Figure 4.7.

4.4 Post-Processing NDD20

NDD20 consists of large scale panoramic images, similar to the Zanzibar data before post-processing. As a result, there is a high amount of noise present in the images which should be ignored and removed before the dataset is utilised for individual identification. Whilst Chapter 5 will focus on both the theory behind, and implementation of, a model capable of

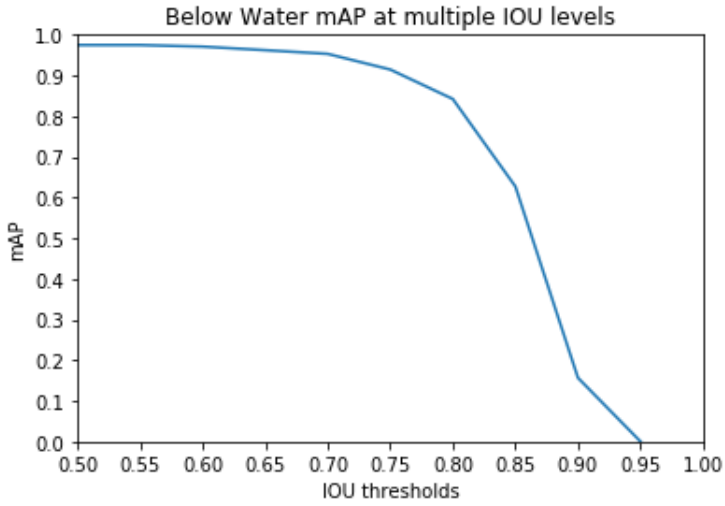

Figure 4.7 Differing mAP@IOU values for the best performing instance segmentation model using the below water data.

individual identification, this Section will detail the processing of NDD20 into a form usable for the task of individual identification.

Utilising the post-processing techniques as outlined in Section 3.5, the detections from the Mask R-CNN model were used to generate a dataset of images to train another model capable of individual identification. This dataset, known as *Segmented NDD20*, is a collection of images which contains only the segmented masks from NDD20.

Once images of the segmentations had been created these were then processed further to create a folder structure which allowed for easy training of an identification model. To facilitate this, each segmentation was checked against the ground truth for the image it was produced from. Any images which contained a dorsal fin with identifying information were placed in a directory containing other examples of that individual. Any fins that did not contain identifying information were removed as their identity could not be guaranteed. Noise which had passed through the mask post-processing were included in a `noise` directory with the goal of allowing any future model to learn how to identify erroneous masks which have made it through post-processing.

Example images from Segmented NDD20 are shown in Figure 4.8. As can be seen there is low inter-class but high intra-class differences. For example there is relatively little difference between the images shown for individuals `32` and `39`, whilst there is a large difference between the three example images for individual `11`. It is this variance and fine-grained nature which makes the task of automatic individual photo-identification particularly challenging.

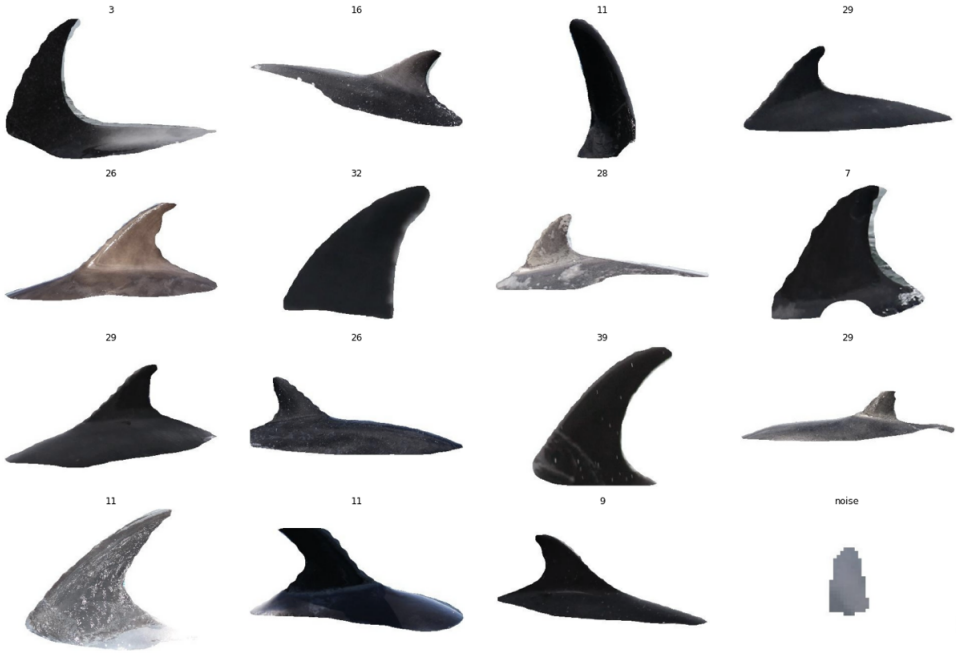

Figure 4.8 Example images from Segmented NDD20. The individual class label is displayed above each image.

In total, Segmented NDD20 contains 1243 images representing 43 classes including `noise`. Individuals `6` and `27` were removed from the dataset by the post-processing algorithm. Upon examination, images of these individuals were captured from extreme angles or with large amounts of splash, making it difficult for the detector to produce accurate segmentations. The dataset is highly skewed, as seen in Figure 4.9 (blue), with approximately 66% of images in the dataset labelled as `noise`. Classes representing individual animals contain a non-uniform number of example images (median = 8.5) varying between 33 examples for individual `11` to just one for individuals `5` and `35`. As such this dataset represents not just a fine-grained but also a few-shot problem, a combination which is extremely challenging for current computer vision methods.

4.5 Additional External Data

After post-processing the data collected during fieldwork into the Segmented NDD20 dataset, it was clear from the class distribution present that it may be beneficial to increase the number of examples per individual, providing a photo-id model more data to learn from. As it was not possible due to sea state conditions to continue data collection in the Coquet to St. Mary's MCZ after 10/10/2019, work was undertaken to procure data from other external sources.

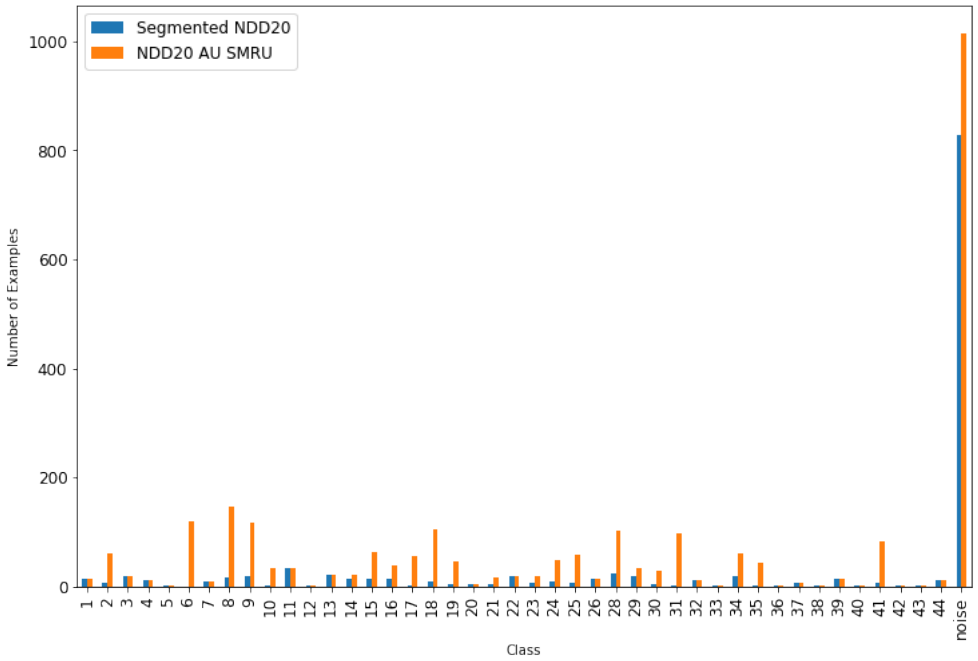

Figure 4.9 The `ID` class label distribution for Segmented NDD20 and NDD20 AU SMRU.

As a result of the large range traversed by cetaceans [186] there was a chance that the individuals catalogued in the MCZ had also been recorded in other areas. The underwater habitat found in the MCZ extends upwards as far as the Moray Firth in Scotland. This, along with the knowledge the animals prefer colder waters, led to the assumption that the animals found in Northumberland would likely also be found in catalogues maintained further north. Examination of photo-id catalogues from the University of Aberdeen, provided after discussions with manager of the University's long-term bottlenose dolphin studies manager Dr. Barbara Cheney, confirmed a catalogue overlap. It was determined that 23 individuals (all bottlenose dolphins) were present in both catalogues. As a result of this analysis, Dr. Cheney provided 1827 additional images of the overlapping individuals to complement Segmented NDD20. Images provided were collected by both the University of Aberdeen and the University of St. Andrews' Sea Mammal Research Unit (SMRU) with the approval of Dr. Monica Arso Civil. These images were captured during surveys undertaken between 2003-2019 and were of the highest quality rating on the scale used by the institutions. All images contained only a single individual.

The additional external data procured was passed through the Mask R-CNN detector and post-processing algorithm, producing images similar to those already present in Segmented NDD20. The post-processed Aberdeen and SMRU data was combined with Segmented NDD20 to produce *NDD20 AU SMRU*. This dataset consists of 2626 images, 1383 more

than Segmented NDD20, representing 44 classes including `noise`. Individual `6`, whilst not present in Segmented NDD20 is accounted for in NDD20 AU SMRU, however individual `27` still remains absent. Like Segmented NDD20, NDD20 AU SMRU is heavily skewed towards `noise` as seen in Figure 4.9 (orange) with 61% of images labelled as this class. Non-`noise` classes however now have a median of 22 examples per class, providing a larger number of example images to train an automatic photo-id model from.

The additional external data procured was passed through the Mask R-CNN detector and post-processing algorithm, producing images similar to those already present in Segmented NDD20. The post-processed Aberdeen and SMRU data was combined with Segmented NDD20 to produce *NDD20 AU SMRU*. This dataset consists of 2626 images, 1383 more than Segmented NDD20, representing 44 classes including `noise`. Individual `6`, whilst not present in Segmented NDD20, is accounted for in NDD20 AU SMRU however individual `27` still remains absent. Like Segmented NDD20, NDD20 AU SMRU is heavily skewed towards `noise` as seen in Figure 4.9 (orange), with 61% of images labelled as this class. Non-`noise` classes however now have a median of 22 examples per class, providing a larger number of example images to train an automatic photo-id model from.

4.6 Summary

This Chapter discusses the collection of bottlenose and white-beaked dolphin abundance estimate data in the coastal waters which make up the Coquet to St. Mary's MCZ. The data, collected over a three month period during the Summer of 2019, was key for testing the Mask R-CNN based detector created in Chapter 3. Image data collected during the surveys provided evidence to suggest that the fin detection system is invariant to changes in geography, time, and species of interest. Performance of the Mask R-CNN model on the data from Northumberland was not hindered, resulting in a mAP@IOU[0.5, 0.75] = [0.96, 0.83]. This was achieved without re-training or fine-tuning the model for use with the Northumberland data.

Further, this Chapter examines the creation of the Northumberland Dolphin Dataset 2020 (NDD20), a fine-grain few-shot computer vision dataset created using the imagery collected during the aforementioned abundance estimate surveys. This dataset is comparable in size to other computer vision datasets aimed at individual animal identification as outlined in Table 4.1 however is unique in the fact that it covers multiple species (both bottlenose and white-beaked dolphins) and provides the ability to identify individuals both above and below the waterline. The usefulness of NDD20 is highlighted through its inclusion in a publicly

hosted Kaggle competition[7] and acceptance to the 7th Fine-Grained Visual Categorization Workshop (FGVC7) hosted at CVPR 2020 [211]. Baseline results on the below water set of NDD20 are also provided.

Finally, the Chapter discusses the use of the post-processing methodology outlined in Chapter 3 for the creation of Segmented NDD20, a dataset to be utilised for the training of a neural network capable of individual identification. A second dataset called NDD20 AU SMRU was created by combining Segmented NDD20 and additional data provided by collaborators at the Universities of Aberdeen and St. Andrews after a cross-catalogue matching process was undertaken, highlighting a 23 individual overlap between the catalogues maintained by Newcastle University and the two collaborating universities.

Thanks to the data collection and dataset creation outlined in this Chapter, evaluation of both the Mask R-CNN based fin detector and post-processing methodology outlined in Chapter 3 has been performed. This evaluation confirms the effectiveness of the model for use in a pipeline of networks to aid in the photo-id process. Furthermore the data collection has both allowed for a cross-catalogue study, indicating a large home range for the bottlenose dolphins which inhabit the MCZ, as well as the creation of a computer vision dataset capable of training a network for individual identification. The creation of said network is examined in detail in Chapter 5.

[7]Kaggle competition: kaggle.com/c/happy-whale-and-dolphin

Chapter 5

Individual Cetacean ID via Automatic Most Likely Catalogue Matching

This Chapter examines the final component in the automatic photo-id pipeline, focussing on individual identification. The component takes as input photo-id catalogue images which have been passed through the dorsal fin detector and post-processing methodology outlined in Chapter 3 to produce a list of most likely catalogue matches. It is important to note here that this component does not intend to replace photo-id researchers by performing the job for them. Instead, the component aims to vastly reduce the search space the researcher needs to examine in order to verify a catalogue match; the component suggests a list of most likely catalogue matches, but ultimately the final decision lies with the researcher.

Beginning by outlining the requirements an automatic system for most likely catalogue matching must meet, the Chapter then discusses possible approaches to the problem and justification for the selected approach. System development is discussed in detail, using the NDD AU SMRU dataset created in Chapter 4 for training and evaluation. Discussion of further processing techniques and their effect on most likely catalogue matching accuracy is discussed, alongside the current limitations of the approach.

5.1 Most Likely Catalogue Matching System Requirements

Before development can begin it is important to outline the requirements of a system capable of most likely catalogue matching. Unlike the detector which could be considered a coarse-grain task, identification of individual cetaceans is an extreme fine-grain problem as they are distinguished from each other using small prominent markings present on the dorsal fin. As the animals are free roaming, there can be high variation in how the fin is captured in the image, discussed in greater detail in Section 3.1.1. This can lead to photo-id catalogues

Figure 5.1 An example post-processed crop which contains some misclassified noise.

with low inter-class but high intra-class differences between the individuals present, seen in Figure 4.8. As a result of this, any system capable of accurate catalogue matching must be able to recognise these minute differences between individuals even when there is high variation in the examples for each individual class.

The system must also be capable of operating using all information provided to it. Other photo-id aides which perform most likely catalogue matching such as finFindR [207] operate using only the trailing edge of the fin, with matching performed using notches and shape. This misses other prominent markings such as long term scarring or pigmentation, as well as the shape of other fin edges. As such, it may be the case that finFindR struggles when operating over a catalogue with few to no notches. To avoid this issue, the system developed must be capable of matching using all available prominent markings.

Further, the system must also be capable of performing accurate catalogue matching under the presence of noise, both classified and misclassified. Datasets developed for the training of this system such as NDD AU SMRU contain a `noise` class which encapsulates all detected mask components which are erroneously retained after post-processing has been applied. This class has extremely high intra-class variance, however it is imperative the system is able to match erroneous components to it. Misclassified noise is defined as that which has been passed downstream as a result of being attached to a valid individual detection mask. In Figure 5.1 for example, the swell captured in the post-processed crop would be considered misclassified noise. Any system performing automatic most likely catalogue matching must be resistant to small amounts of misclassified noise in order to produce accurate identity suggestions.

Any developed system must also be capable of handling examples of individuals which are not present in the photo-id catalogue. Due to the free roaming nature of cetaceans (or indeed any wild animal) and the limitations on photo-id survey size dictated by both weather and workforce, there is no guarantee that every animal who makes use of the survey area will

be captured. New animals may also become resident in the area through birth or migration. When these animals are eventually captured during a survey and their image processed, the system must be capable of recognising this as an individual not currently present in the catalogue and highlight this to the researcher. This is made more difficult given the extreme fine-grain nature of the catalogues. As a result, this requirement necessitates the system must be capable of recognising uncertainty or understand a notion of similarity between an input and the class examples present in the catalogue.

In traditional computer vision classification models if the model was required to classify a new class, this would require a large number of example images as well as model retraining or fine-tuning. However, cetacean researchers are highly unlikely to possess a large number of example images for the new individual from first encounter. As such, the system must be adaptable enough so as to not require extensive retraining when new individuals are added to the catalogue.

5.2 Possible Approaches

Out of all requirements an automatic most likely catalogue matching system must meet, arguably the most important is the need for flagging of previously unseen individuals. As noted, this necessitates any underlying computer vision model to have some notion of uncertainty or similarity. It is this requirement that guided approach selection.

5.2.1 Bayesian Dropout

Traditional computer vision classification models do not meet this requirement. If an example image of a new individual was seen by a traditional CNN trained on a photo-id catalogue dataset, this model would still attempt to provide a classification based on the classes present at train time. As deep computer vision models operate on point estimations of parameters, unlike Gaussian processes where the probability distribution is defined over a function, this removes the ability to produce helpful indicators of uncertainty such as prediction confidence bounds [72].

One way to create a notion of uncertainty from this is through the use of Bayesian dropout. Vanilla dropout has found widespread use in the training of generalisable deep learning models. At train time, nodes in the model are intentionally not updated during a training step with some probability, usually defined as a hyperparameter with the goal of aiding model generalisability [196]. At test time, no dropout is performed and all nodes are utilised for the prediction.

Bayesian dropout re-frames this technique by also performing dropout at test time, again with some hyperparameter defined probability [73]. For each classification output, the model performs inference some large N number of times. During each run model nodes are randomly disabled, zeroing out their weight and effecting the ability of the model to produce a prediction. By performing this multiple times and producing N classifications, a probabilistic distribution is determined which can be used to understand the uncertainty of the model. A final overall prediction is generated by taking the mean of all N predictions used to generate the probability distribution. If randomly dropping nodes at test time results in the model producing a wide variety of outputs, resulting in a diffused probability distribution, this suggests the model is uncertain; the lower the variance of the probability distribution, the more certain the model is.

Whilst Bayesian dropout has found use in areas such as time-series forecasting [120], widespread use has not been adopted in areas such as computer vision despite attempts [110]. This can be contributed to recognised issues such as ill-defined variational objective, the use of improper priors, and the potential for clarity issues between model uncertainty and risk [102, 159]. Further to this, the computational expense of performing Bayesian dropout is large given the need for multiple inferences required to produce the classification probability distribution.

This is the main reason why Bayesian dropout was not utilised for automated most likely catalogue matching. Should a researcher wish to process a large batch of images after a field survey for example, the need for multiple inferences would vastly inflate the time required for the batch operation to complete. Issues also arise meeting other system requirements. Even when using Bayesian dropout, the underlying model would still require retraining or fine-tuning to output newly catalogued individuals.

5.2.2 Siamese Neural Networks

Rather than producing a classification and measuring uncertainty as is the case with Bayesian dropout, Siamese Neural Networks (SNNs) aim to incorporate the notion of similarity into the model. This is achieved by connecting two or more identical CNNs in parallel, each sharing the same backbone architecture, initial and updated weights, and hyperparameters. Each CNN in the SNN is designed to produce an embedding, or a d-dimensional representation, of the input. The size of this embedding is set via hyperparameter and dictates how many d dimensions the output of the SNN will be. For example, if an SNN is created with an embedding size of 10, each CNN may take a high dimensional input of size *width * height * channels* and output a 10-dimensional embedding, a float vector of size 10, which represents the input image. A visualisation of a two branch SNN can be seen in Figure 5.2.

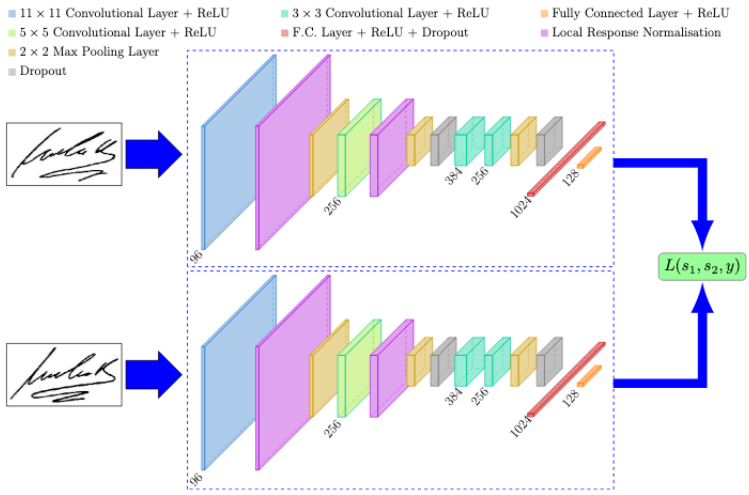

Figure 5.2 An example SNN architecture for signature verification. Image from [59].

At train time, each CNN branch receives a different image and generates an embedding. These embeddings are compared to one another in order to optimise some loss function. By optimising in such a way that input images of the same class have similar embeddings but those of different classes are dissimilar, the SNN can be tuned to provide a measure of image similarity. Once trained only one branch of the model is retained. This allows a single image to be embedded by the model which can then be compared to the training examples.

It is this ability which has resulted in the wide use of SNNs for verification or identification problems in computer vision [59, 226]. Specifically in conservation tech, SNNs have found use in fine-grain species identification problems [10, 220] as well as in more extreme fine-grain individual animal identification [51].

Clustering Embeddings in a Latent Space

By storing the embeddings generated for each trained class it is possible to produce a list of likely class predictions for a new image by measuring the Euclidean distance between the generated embedding and those previously produced when plotted into some d-dimensional latent space. If the SNN has trained in such a way as to produce low intra-class, high inter-class difference between generated embeddings then this will create class clusters when plotted in the latent space.

An example of this behaviour can be seen in Figure 5.3 which shows a 2-dimensional visualisation, produced using Principle Component Analysis (PCA), of the embedding locations for a subset of the MNIST dataset [122]. Here, an SNN has been trained for 100

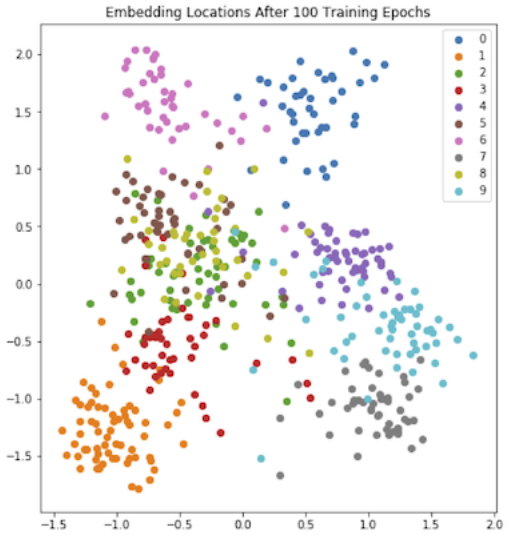

Figure 5.3 A 2-dimensional visualisation of a multi-dimensional latent space produced by an SNN trained on the MNIST dataset [122] for 100 epochs.

epochs to generate embeddings of images for the 10 unique classes. As can be seen, the model is able to generate embeddings in such as way as to cluster those of the same class in the latent space. Note that some clusters are visualised on top of each other due to the dimensionality reduction performed in order to show the latent space on the page.

It is important to note here that the value of the embeddings is not necessarily important, just the distances between them. Notice how all points in Figure 5.3 lie within approximately -1.5, 2.0 on the x-axis and -2.0, 2.0 on the y-axis. There is nothing inherently good or bad about an SNN that embeds within this range, all that matters is the points are clustering in their respective classes.

Meeting the Outlined Requirements

When compared to Bayesian dropout, the computational expense of performing inference with an SNN is quite small. Whilst training requires the use of a branched CNN architecture in order to optimise the loss function, this is reduced to just one branch at inference time. Generating a list of most likely catalogue matches would only require an image to be passed through the network once in order to generate an embedding, and similarity via Euclidean distance measurement in the latent space is cheap to perform. As such, producing a list of most likely catalogue matches is overall more computationally efficient using SNNs as opposed to Bayesian dropout.

The clustering of class embeddings in the latent space also allows for easy identification of potential previously unseen individuals. Passing the dorsal fin of an individual not present at train-time through the model would result in, theoretically, a distinct embedding which would plot into a unique point in the latent space far from any existing class clusters. By implementing a threshold on the Euclidean distance measurement, potentially unseen individuals could be easily flagged to the researcher for further investigation. Clustering also removes the need for re-training to allow for matching to previously unseen individuals when they are added to the catalogue. Adding a new class to the latent space can be achieved simply by defining embeddings to a new class cluster and including these in future distance measurements.

In addition, SNNs are capable of operating over all information provided to them. This can be achieved by not limiting the embedding generation to one specific part of the dorsal fin. It also stands to reason that this embedding generation will be robust enough to deal with small amounts of retained background noise given a high enough number of dimensions. Overall, the use of SNNs for most likely catalogue matching far outweighs the use of Bayesian dropout. It is for this reason the decision was taken to first begin development of a model capable of most likely catalogue matching using SNNs.

5.3 Siamese Neural Network Background

As this work will focus on utilising SNNs for the task of most likely catalogue matching it is important to first outline some key concepts. This Section provides the required background knowledge for various terms which are used when discussing SNN development and evaluation later in this Chapter.

5.3.1 Pairwise vs Triplet Ranking Loss

Training of any neural network is performed through the optimisation of a loss function. For SNNs, a group of loss functions known as Ranking Losses are utilised. Here, the goal is not to predict a class label but rather a distance between model inputs. As such, they are perfect for training SNNs.

During training an SNN will generate embeddings for some received inputs and generate a similarity value (e.g. via Euclidean distance when plotted into a latent space). This similarity value is then used to optimise the Ranking Loss which in turn tells the model how to adjust to create better embeddings, for example how to bring two embeddings closer when they are of the same class. The type of Ranking Loss utilised for training and the number of branches

present in the SNN are intertwined. Two of the most commonly used Ranking Losses are Pairwise Ranking Loss and Triplet Ranking Loss.

Pairwise Ranking Loss

SNNs which make use of two branches can be optimised using a Pairwise Ranking Loss, a visualisation of which can be seen in Figure 5.4. Here the model is trained using data points made up of two inputs. The first input is called the Anchor, which defines the class the model is training to optimise for. The second input can be either a Positive containing another example of the Anchor class, or a Negative containing an example of some class other than the Anchor.

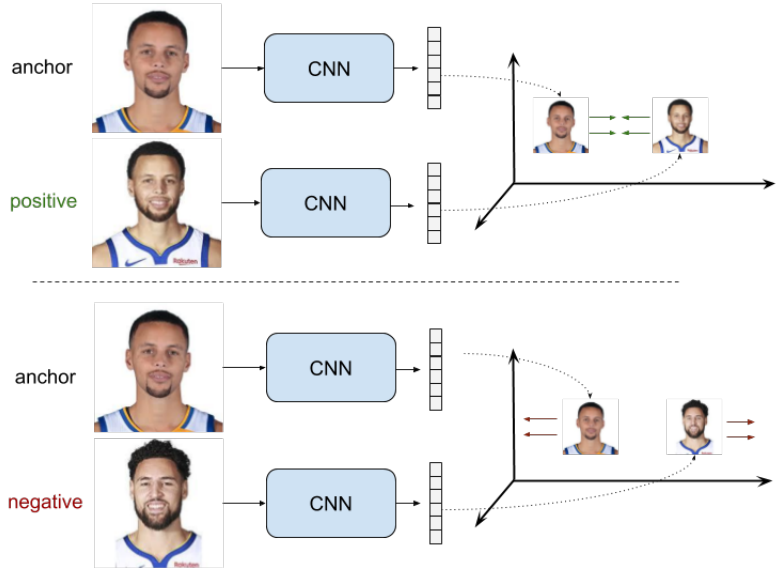

Figure 5.4 SNN optimisation using Pairwise Ranking Loss. Image from [86].

Using these two input types Pairwise Ranking Loss can be used to optimise in such a way that the model learns to produce embeddings with a small distance between Anchors and Positives, and a large distance between Anchors and Negatives. Mathematically Pairwise Ranking Loss can be defined using Equation 5.1:

$$L = \begin{cases} D(A,P) & \text{if Positive Pair} \\ max(0, m - D(A,N)) & \text{if Negative Pair} \end{cases} \tag{5.1}$$

Where L is the loss, $D(A,P)$ is the distance between the Anchor and the Positive, and $D(A,N)$ is the distance between the Anchor and the Negative. When optimising for Positive Pairs the loss function will only ever return 0 when the distance between the Anchor and the

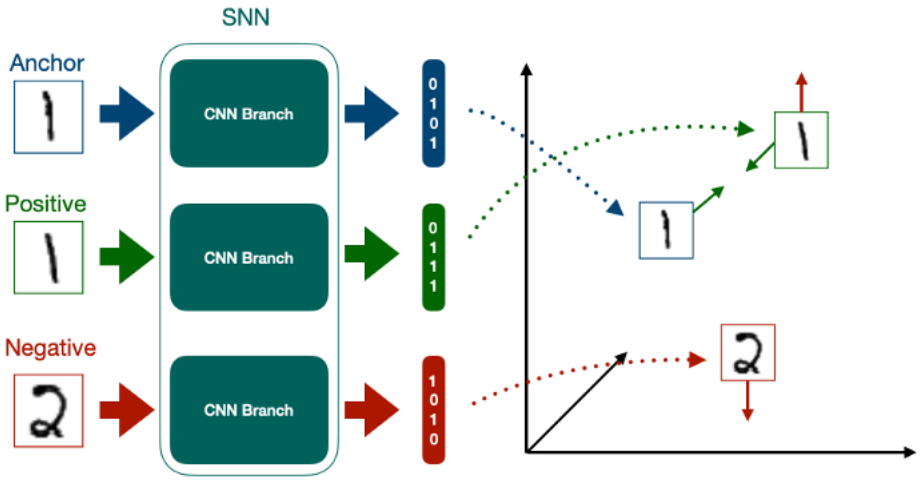

Figure 5.5 SNN optimisation using Triplet Ranking Loss. Each input is passed to a branch of the SNN and an embedding is produced. These embeddings are used to optimise future embedding generation, aiming to pull the Anchor and Positive together whilst pushing the Positive and Negative apart. Example Anchor, Positive, and Negative from the MNIST dataset [122].

Positive is 0, ensuring these embeddings are nearly always pulled closer. When optimising for Negative Pairs, the loss function will return 0 when the distance between the Anchor and the Negative is greater than some margin m. As such, a weight update is not performed when the distance between the Anchor and the Negative is sufficiently large.

Triplet Ranking Loss

One of the main problems presented by Pairwise Ranking Loss is the issue of model collapse, occurring after a large amount of Positive Pair optimisations. In this scenario, the distance between Anchors and Positives are pushed so close together in the latent space as to produce the same embedding. This can in turn affect the model's ability to understand variation in input and similarity scoring.

Triplet Ranking Loss aims to avoid this issue by training on triplets of data points rather than pairs, with each triplet containing an Anchor, a Positive, and a Negative. SNNs which make use of Triplet Ranking Loss are often named Triplet Networks in literature (such as in Hoffer *et al.* [99] which first made use of them), however the only difference between the structure of an SNN using Pairwise Ranking Loss or Triplet Ranking Loss is the number of branches - two or three respectively.

Just like with Pairwise Ranking Loss, Triplet Ranking Loss takes as input the generated embeddings for each branch and optimises to pull the Anchor and Positive close whilst

pushing the Negative away, as visualised in Figure 5.5. Optimisation is performed using Equation 5.2:

$$L = max(0, D(A,P) - D(A,N) + m) \tag{5.2}$$

By utilising a triplet, the loss function evaluates to 0 when $D(A,N) > D(A,P) + m$. This occurs only when the triplet contains examples the model is already well trained on and no further optimisations can be gained. By enforcing m, where typically $m = 0.2$ thanks to work by Schroff *et al.* [185], the function ensures embedding variation between distinct inputs thus allowing for a similarity score to be computed between the Anchor and the Positive in all cases. Thanks to the advantages of Triplet Ranking Loss over its Pairwise counterpart, the decision was made to make use of this loss function and train an SNN with three branches. Further, Triplet Ranking Loss has been shown to perform well on individual identification tasks in both humans [97] and animals [220], providing evidence to support its use for training a most likely catalogue matcher.

5.3.2 Semi-Hard Triplet Mining

When training a model, care should be taken to ensure learning occurs at every step. When using Triplet Ranking Loss however, learning does not occur during training steps where the loss evaluates to 0, such as when $D(A,N) > D(A,P) + m$. Negatives provided should be sufficiently difficult such that the triplet allows the loss to evaluate to a non-zero value, allowing the model to learn. However care should also be taken so as to not provide the model with triplets that are too difficult, as this will increase optimisation and thus overall training time.

This leads to somewhat of a Goldilocks problem. Triplets must be not too soft to prevent learning, but not too hard to dramatically increase training time. Semi-Hard Triplet Mining aims to fix this problem, providing triplets which are *just right*. First, three types of triplets are defined:

- **Easy**: where $D(A,P) + m < D(A,N)$
- **Hard**: where $D(A,N) < D(A,P)$
- **Semi-Hard**: where $D(A,P) < D(A,N) < D(A,P) + m$

The goal of Semi-Hard Triplet Mining is to locate as many Semi-Hard triplets from the training set as possible. These are triplets whereby the loss still evaluates to a positive value however the Anchor is closer to the Positive than the Negative when plotted in the latent

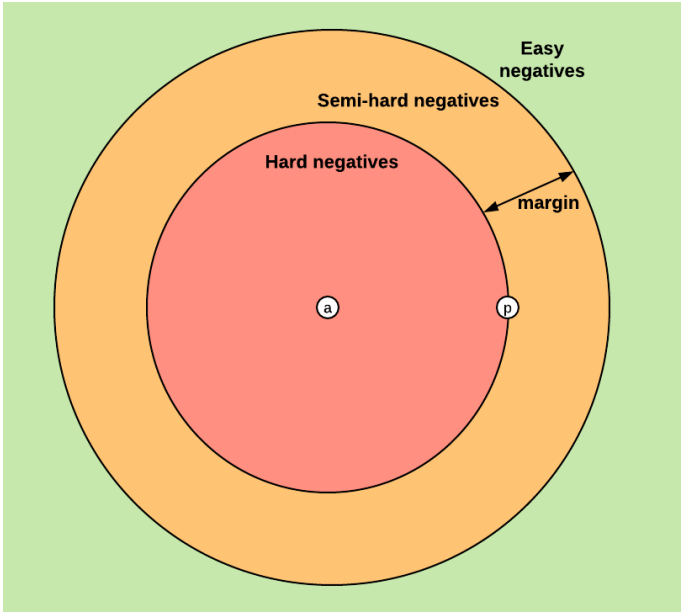

Figure 5.6 A visualisation of the areas in the latent space where Easy, Hard, and Semi-Hard triplets can occur, where a is the location of the Anchor and p is the location of the Positive. Image from [148].

space, as seen in Figure 5.6. This allows for fast training whilst still providing enough triplet difficulty for the model to learn during training.

Finding, or mining, Semi-Hard triplets can be performed either Offline or Online. In Offline mining, the entire training set is converted into triplets before the training epoch occurs and those which fit the Semi-Hard definition are utilised. With Online mining, Semi-Hard triplets are generated on the fly as required. Generally, Online mining results in faster training when compared to Offline mining as this allows for the ability to update our definition of a Semi-Hard Triplet as training progresses.

5.3.3 Class Prototyping

After SNN training it is possible to obtain likely classifications for an input based on Euclidean distance measurements between the input's embedding and the previously generated embeddings when plotted into the latent space. If there is a large number of embeddings in the space however this can increase classification time, as the input's embedding must be checked against every other in the space.

There are ways to reduce the time taken for this calculation to complete by reducing the number of distance measurements which occur. A naive approach would, for example, be to randomly select one embedding for each class and measure the distance between it and the

input embedding such that the distance to each class is only measured once, vastly reducing the computation required for classification. This may only work however when the class embeddings are perfectly clustered in the latent space, which will likely not be the case when using real world data.

As no neural network is perfect there may be cases where embeddings are not clustered with their class, such as in Figure 5.7 where an embedding of class `cross` has been generated such that it is surrounded by example of class `square` - far from the other `cross` examples. There is also a triangle in the top-right of the Figure which represents the embedding location of an unclassified inference image.

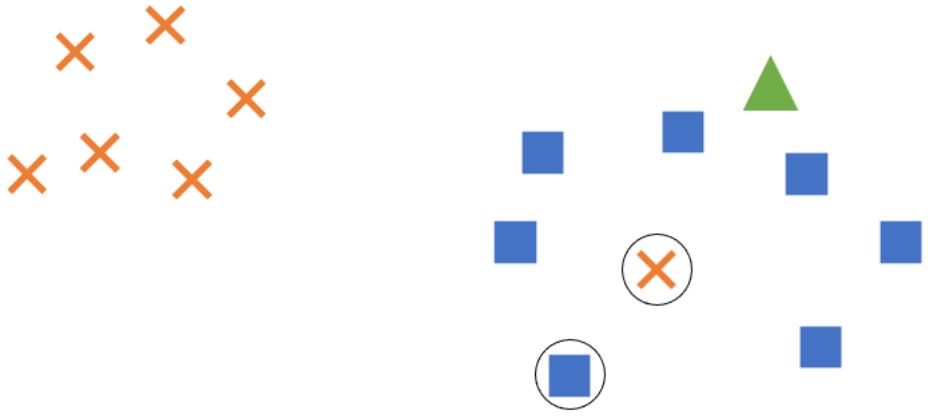

Figure 5.7 An example latent space with two classes (`cross` and `square`) alongside a triangle which represents the embedding location of an unclassified inference image. The two class examples selected for distance measurement to classify the triangle using the naive approach are circled.

Using the naive approach to classify this triangle as either a `cross` or `square`, assuming the two randomly selected class embeddings are the ones circled in the Figure, then the triangle would be classified as an example of class `cross`. However, looking at the space globally is is clear the triangle should more likely be classified as a `square`; the chosen `cross` is simply an outlier. By selecting embeddings to measure from, the risk of outliers skewing the distance measurement, and thus the final classification, increases.

This risk can be mitigated through the use of class prototypes, generalised embeddings generated from the example embeddings for each class. By making use of prototypes, the effect of outliers during classification is reduced. These can be calculated using multiple different methods, however simple techniques such as defining the prototype as the median embedding for all class examples works well.

Figure 5.8 shows the same example two class latent space as previously, however it now also displays the generated class prototypes P_x and $P_■$ respectively. If the distance measurement is performed using the prototypes, the triangle is now classified as a `square`, which is more likely given the construction of the global space.

Figure 5.8 An example latent space with two classes (`cross` and `square`) alongside a triangle which represents the embedding location of an unclassified inference image. The two class prototypes used to classify the triangle are circled.

This method is not without its limitations either however. If all examples of the `cross` class formed a circle of radius 1 around the origin and all `square` examples formed a circle of radius 2, in both cases the class prototypes would be formed at the origin resulting in equal distance measurements. Although the chances of this are small, and these formations could be avoided using different hyperparameters or model architecture, it is important to be aware that this could happen when utilising prototypes.

5.3.4 Top-N Accuracy

In coarse-grain computer vision tasks such as image classification, the effectiveness of a model is evaluated using, among other metrics, an accuracy score. Given an image in the test dataset, the model's class prediction for the image, defined as the class which has the highest probability assigned to it in the model's final softmax layer, is compared against the ground truth class label. If the predicted and ground truth classes are the same, the model is correct and is operating as intended. Performing this process iteratively over all images in the test

dataset provides an accuracy score, often written as a percentage denoting how many test images the model classified correctly.

Due to the low inter-class differences between the classes in a fine-grain dataset however, the same model's softmax layer may struggle to consistently assign the highest probability to an image's ground truth. Using the coarse-grain definition of accuracy, this model may now perform poorly. However for certain tasks, such as most likely catalogue matching, the model may be considered effective simply if it is able to reduce the range of classification possibilities.

As such, the top-N accuracy metric is often utilised for fine-grain classification tasks [36, 75, 242]. Rather than only using the softmax's highest probability, the model instead takes the N highest probabilities, outputting its prediction as a list of possible values. If the ground truth class is contained within the list, the model is considered to be correct.

For example, utilising top-10 accuracy the model would output the 10 highest probabilities for an image and would be considered correct if the ground truth label was within this list. Utilising top-1 accuracy would force the model to output a single prediction, which would be the same as using the coarse-grain accuracy definition.

As the task of most likely catalogue matching is a fine-grain problem, developed SNNs are evaluated using the Top-N Accuracy metric. Furthermore, as the goal of this work is to produce a system which aides researchers through the task of catalogue matching rather than fully replace them, it is beneficial for the SNNs to produce a list of predictions as this will greatly reduce the number of individuals the researcher needs to examine in order to be confident of a match.

5.4 Siamese Neural Network Development

Before an SNN capable of most likely catalogue matching could be created, a development environment was first required to be selected. Initial testing first began using the Tensorflow framework [2], specifically version 1.14, as this would keep consistency with the dorsal fin detector developed in Chapter 3.

Replication of the work undertaken by Vetrova *et al.* [220] for moth species identification proved promising. However, expanding to individual identification using the NDD AU SMRU dataset faltered when using Tensorflow, believed to be caused by a bug[1] in how the Keras backend handles Batch Normalisation for models with multiple inputs resulting in model collapse, a phenomenon which causes the model to return the exact same output

[1]Batch Normalisation Issues in Keras/Tensorflow: see github.com/keras-team/keras/issues/11927 and github.com/keras-team/keras/issues/9498

embedding regardless of input image. As such, the decision was made to switch to using the PyTorch framework [161] which does not suffer these issues. To aid development, this work made use of Adam Bielski's PyTorch implementation of SNNs[2].

Two backbone architectures were tested during SNN development. The first of these was the architecture defined in Vetrova *et al.* [220], hereafter denoted as *VarvaraNet*. This was utilised to examine if a network which is proven capable at species identification is also able to perform well for the task of individual identification. The second was a custom architecture consisting of a Convolutional layer, a Dropout layer, a PReLU layer, and a MaxPool layer (stride = 2). This network was utilised to examine if a more basic backbone would be capable of performing well given the fine-grain, few-shot nature of the task. This architecture is hereafter denoted as *EmbeddingNet*.

5.4.1 Hyperparameter Tuning Via Bayesian Optimisation

Like all models, SNNs have multiple hyperparameters which must be tuned. As such, work began to select which hyperparameters should be tuned and how. Since developing the Mask R-CNN fin detector, discussed in Chapter 3, the area of hyperparameter optimisation has advanced considerably. Multiple frameworks now exist which take a Bayesian approach to finding the optimal hyperparameter values. Unlike optimisation through a Grid Search whereby all combinations of user-defined hyperparameter values are evaluated (see Section 3.4.3 for an example of this), with Bayesian Optimisation the user only needs to define the upper and lower bounds for each hyperparameter. The search space is then explored using a probabilistic methodology, locating the optimal set of hyperparameters within the ranges provided. This speeds up the optimisation process as values unlikely to yield promising results are ignored. As such, a larger number of hyperparameters can be optimised when compared to a Grid Search.

The Optuna framework [3] was utilised for hyperparameter optimisation. Whilst Optuna allows users to make use of custom optimisation algorithms, this work specifically made use of the built-in Tree-structured Parzen Estimator (TPE) algorithm. Optuna performs optimisation iteratively. This means that, for each iteration and for each hyperparameter, TPE fits one Gaussian Mixture Model to the set of hyperparameter values, x, associated with the current optimal values, $l(x)$, and another to the remaining hyperparameter values, $g(x)$. Optimal values for each iteration are selected by maximising the ratio $l(x)/g(x)$, with the final trial producing the current optimal hyperparameter values. For a more in-depth discussion of TPE, see Bergstra *et al.* [25].

[2] *Siamese and triplet learning with online pair/triplet mining* repository by Adam Bielski: github.com/adambielski/siamese-triplet

Through Optuna, TPE was utilised to set the learning rate to a `log uniform` value between 1e-6 and 1e-3, for use with either the SGD or Adam optimiser. Weight decay was set to a `log uniform` value between 1e-6 and 1e-1. Step size was set to an `int` value between 5 and 10, with the γ value for this set to a `log uniform` between 1e-3 and 1e-1. The margin m defined in the Triplet Ranking Loss (see Equation 5.2) was set to a `log uniform` value between 0.1 and 1.0. The final embedding layer was tuned to produce an `int` value between 16 and 128.

Optimisation of the number of network blocks was also examined. For VarvaraNet a block consisted of a Convolutional Layer, a MaxPool layer (stride = 2), a ReLU layer, and a Dropout layer. For EmbeddingNet a block consisted of a Convolutional layer, a Dropout later, a PReLU layer, and a MaxPool layer (stride = 2). During searching, the number of blocks was treated as a hyperparameter optimising for an `int` between 1 and 5 blocks. The size of the initial Convolutional layer was also tuned, searching for an optimal `int` value between 16 and 100. Subsequent layers were double the size of the previous. Dropout was set to search for a `log uniform` value between 0.1 and 0.7. The kernel size of the initial Convolutional layer was set to a `categorical` value of either 5, 6, 7, or 8 with subsequent layers set according to $max(1, k-2)$ where k is the kernel size of the previous Convolutional layer.

5.4.2 Data Augmentation Strategy

The use of data augmentation was also examined. The decision was made to reduce the variety of augmentations performed compared to Mask R-CNN development, as discussed in Section 3.3.4. At this stage in the pipeline the data seen by the SNN has been post-processed, thus it would not be realistic to utilise an aggressive data augmentation strategy over the NDD AU SMRU dataset. Further, an aggressive strategy may obscure the identifying markers present on the fins too much for meaningful training to occur.

The first strategy, *Colour Jitter*, randomly perturbs the input images' brightness by a factor of between 0.8 and 1.2, contrast by a factor of between 0.8 and 1.2, saturation by a factor of 0.9 and 1.1, and hue by a factor of -0.1 and 0.1. The second, *Perspective Shift*, randomly distorts the input image's perspective by a factor of 0.5. The third, *Greyscale*, converted the three-channel RGB input image into a single-channel greyscale image. Tests examining combinations of these strategies were also examined, such as augmenting with both Colour Jitter and Perspective Shift. Note that Greyscale cannot be combined with Colour Jitter due to the reduction in colour channels required.

5.5 Siamese Neural Network Model Selection

Models with both VarvaraNet and EmbeddingNet architectures were trained for the task of most likely catalogue matching using the NDD AU SMRU dataset and the data augmentation strategies defined in Section 5.4.2. Hyperparameter optimisation was performed for each architecture-augmentation combination, as each architecture and augmentation strategy may influence the optimal hyperparameters for the model.

To perform hyperparameter optimisation with Optuna both a train and validation set is required, with the former utilised to train the selected architecture using the current iteration's selected hyperparameters and the latter utilised to evaluate how well this model performs against unseen data - acting like the test set for each iteration. As such the dataset was first divided using an 80-20 train-test split. The train set was then divided further for optimisation, with 30% of the train set held for validation, resulting in an overall 56-24-20 train-validation-test split.

Whilst the train and validation splits may feel unnatural (a 56-24 split is not common in literature), splitting in this way ensures that a high variety of Semi-Hard triplets can be generated at all points in the training process. To further aid this, before splitting the dataset was filtered to remove any classes which contained fewer than 6 example images. Performing this step has the added benefit of allowing some individuals to be held back to examine the SNN's ability to flag those it has not been trained to recognise, as these can be treated as uncatalogued individuals.

Once the final optimal hyperparameters had been located using Optuna, the train and validation sets were recombined and used to train the selected architecture from scratch using the located hyperparameters, alongside the selected data augmentation strategy. Once trained, the model was then evaluated using the test set.

Figure 5.9 shows the results of training an SNN to perform most likely catalogue matching on the NDD AU SMRU dataset. Each model trained is evaluated using top-1, top-5, and top-10 accuracies. As can be seen, the best performing model is a VarvaraNet trained without the use of any data augmentation. This model achieves the highest test set accuracy at all evaluated metric thresholds, achieving 40.9% top-1, 68.9% top-5, and 83.1% top-10 accuracies. These results provide evidence that SNNs are capable of fine-grain, few-shot individual level identification. If the model was deployed into production and utilised by cetacean researchers, then these levels of accuracy would vastly reduce the search space required to perform most likely catalogue matching.

On the whole, it is the case that models using an EmbeddingNet backbone perform worse than those using a VarvaraNet backbone, even when utilising the same data augmentation

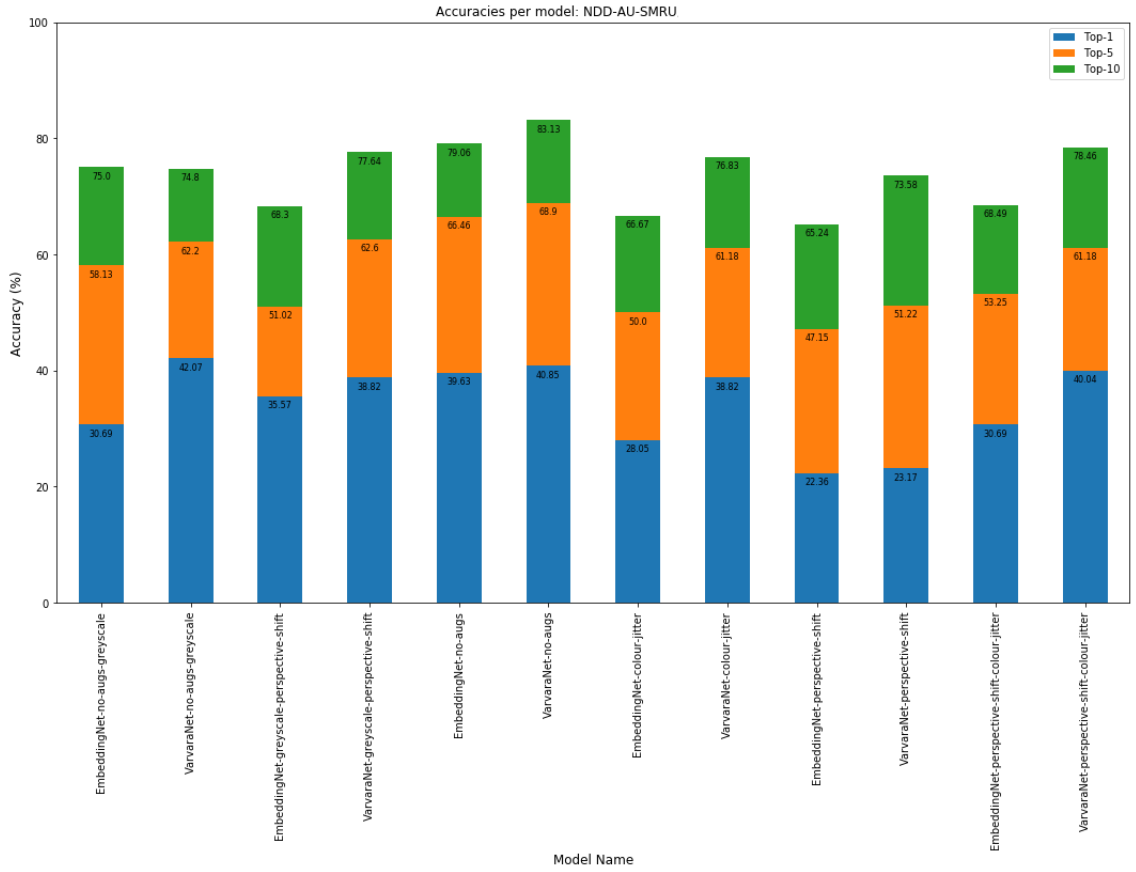

Figure 5.9 Results of SNN training for the task of most likely catalogue matching on the NDD AU SMRU dataset.

strategies. Furthermore it can be seen that, in general, the data augmentation strategy chosen for training has little effect on final model performance against the test set as evidenced by the lack of variation in accuracy metrics between models when trained using different strategies. This is especially interesting with regards to the greyscale augmentation as whilst there has been a drop in accuracy when compared to the best performing model this is only slight. However, this does suggest that a reduction in colour channel leads to some information loss. The fact that the best performing model is one which makes use of no data augmentations may suggest that even strategies which only perturb the input slightly still has a negative effect on the model's ability to extract identifying information.

5.5.1 An Evaluation of Optimal Model Hyperparameters

The optimal SNN hyperparameters chosen through Bayesian optimisation for each architecture-augmentation training run can be seen in Table 5.1. Of the models trained, 83% of them

Model Backbone	Data Augmentation Strategy	Network Blocks	Initial Convolutional Layer Size	Initial Convolutional Layer Kernel Size	Dropout	Learning Rate	Optimiser	Weight Decay	Step Size	γ	Embedding Size	Triplet Ranking Loss Margin
EmbeddingNet	Greyscale	5	37	8	0.677	2.578e-6	Adam	5.008e-5	6	0.048	120	0.551
VarvaraNet	Greyscale	2	60	5	0.358	1.031e-4	Adam	3.680e-5	5	0.036	17	0.863
EmbeddingNet	Greyscale & Perspective Shift	5	26	7	0.261	1.214e-4	Adam	8.364e-3	10	0.081	65	0.684
VarvaraNet	Greyscale & Perspective Shift	5	62	5	0.269	1.817e-6	Adam	1.954e-4	5	0.030	91	0.860
EmbeddingNet	None	3	29	7	0.242	2.835e-5	Adam	9.106e-3	8	0.001	69	0.758
VarvaraNet	None	2	59	6	0.169	7.253e-6	Adam	4.338e-2	10	0.012	106	0.796
EmbeddingNet	Colour Jitter	5	44	6	0.197	1.876e-5	Adam	3.567e-6	6	0.011	40	0.436
VarvaraNet	Colour Jitter	3	38	5	0.684	9.251e-4	SGD	7.512e-4	5	0.004	90	0.281
EmbeddingNet	Perspective Shift	5	42	6	0.120	1.653e-5	Adam	3.256e-3	5	0.014	60	0.273
VarvaraNet	Perspective Shift	1	45	6	0.447	2.890e-4	Adam	4.458e-4	6	0.004	24	0.635
EmbeddingNet	Colour Jitter & Perspective Shift	3	28	6	0.559	1.348e-6	Adam	1.608e-5	5	0.073	51	0.458
VarvaraNet	Colour Jitter & Perspective Shift	2	35	5	0.286	4.093e-4	SGD	5.352e-4	9	0.068	28	0.826

Table 5.1 Optimal SNN hyperparameters for each architecture-augmentation combination located using Optuna over 100 iterations. Results given to 3 decimal places where applicable.

use Adam [113] as an optimiser, including the best performing model (VarvaraNet without data augmentation). This aligns with the belief within deep learning research that Adam will often provide optimal model training [107].

Furthermore, in general best results are achieved when utilising a low probability of dropout. This suggests that the models are not overfitting even with relatively small amounts of training data, which may be due to the low inter-class, high intra-class differences present in the dataset, as seen in Figure 4.8.

Interestingly it is also the case that all hyperparameter optimisation runs, regardless of architecture or data augmentation strategy, settle on a Triplet Ranking Loss margin above 0.2, the value commonly used as a default for this hyperparameter thanks to work by Schroff *et al.* [185]. There is a high deviation in margin value between the models, which may suggest that the use of 0.2 in all cases by default will not lead to optimal results.

One important takeaway from the use of Bayesian optimisation techniques is that for some hyperparameters which require a continuous input, such as learning rate or weight decay, the optimal value may be one which would likely not be selected by a human when performing a Grid Search. This highlights the effectiveness of Bayesian optimisation over a Grid Search as, whilst the difference between a learning rate of 2e-6 selected during a Grid Search and 2.578e-6 selected by Bayesian optimisation, may only be a few percentage points increase in test set accuracy, this still results in an overall more generalisable model.

Appendix

A mAP@IOU[0.5:0.95] Scores for Mask R-CNN Grid Search Models

Model Name	mAP@IOU[x]									
	0.5	**0.55**	**0.6**	**0.65**	**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**
20190909T0723	0.798	0.778	0.770	0.738	0.690	0.644	0.516	0.298	0.099	0.000
20190912T0045	0.780	0.745	0.710	0.663	0.596	0.539	0.425	0.214	0.080	0.000
20190908T2139	0.910	0.894	0.879	0.863	0.797	0.686	0.571	0.349	0.082	0.000
20190908T0202	0.817	0.771	0.757	0.727	0.678	0.594	0.508	0.281	0.071	0.000
20190912T0608	0.910	0.902	0.892	0.861	0.838	0.771	0.642	0.463	0.197	0.000
20191103T1921	0.926	0.905	0.901	0.878	0.836	0.768	0.643	0.360	0.115	0.000
20191102T1051	0.914	0.895	0.864	0.861	0.836	0.812	0.601	0.461	0.132	0.000
20190907T1500	0.837	0.814	0.790	0.780	0.739	0.649	0.497	0.289	0.101	0.000
20190907T0933	0.844	0.805	0.774	0.748	0.709	0.592	0.471	0.243	0.069	0.000
20190908T0957	0.884	0.880	0.863	0.835	0.801	0.681	0.577	0.324	0.048	0.000
20190908T0417	0.827	0.803	0.760	0.736	0.707	0.619	0.485	0.287	0.098	0.000
20190908T1102	0.790	0.775	0.712	0.699	0.650	0.581	0.480	0.256	0.110	0.000
20190908T2043	0.929	0.913	0.904	0.881	0.841	0.804	0.680	0.482	0.126	0.000
20191101T1633	0.909	0.909	0.902	0.883	0.864	0.780	0.662	0.419	0.135	0.000
20190907T2026	0.919	0.919	0.901	0.861	0.843	0.782	0.628	0.412	0.118	0.000
20191103T0044	0.901	0.888	0.863	0.853	0.832	0.736	0.616	0.348	0.090	0.000
20191102T2006	0.919	0.897	0.897	0.865	0.836	0.747	0.583	0.391	0.125	0.016
20191103T1441	0.882	0.873	0.854	0.835	0.763	0.739	0.646	0.401	0.129	0.000
20190907T2126	0.827	0.803	0.778	0.743	0.668	0.544	0.442	0.257	0.075	0.000
20190908T1939	0.811	0.790	0.745	0.703	0.640	0.579	0.434	0.203	0.095	0.000
20190907T0932	0.834	0.805	0.780	0.718	0.649	0.567	0.391	0.239	0.063	0.000
20190907T1451	0.804	0.766	0.724	0.689	0.626	0.560	0.373	0.197	0.054	0.000
20190908T1204	0.901	0.890	0.872	0.855	0.825	0.733	0.674	0.453	0.185	0.000
20190907T2215	0.928	0.917	0.917	0.889	0.855	0.789	0.660	0.452	0.137	0.000
20190905T2336	0.822	0.799	0.751	0.736	0.695	0.640	0.536	0.265	0.116	0.000
20190905T2202	0.848	0.823	0.814	0.773	0.712	0.619	0.555	0.281	0.116	0.000
20190905T1813	0.937	0.917	0.907	0.904	0.851	0.775	0.713	0.452	0.113	0.000
20191102T1528	0.919	0.910	0.910	0.877	0.858	0.778	0.661	0.425	0.112	0.016
20190906T0332	0.902	0.895	0.877	0.867	0.811	0.738	0.630	0.417	0.124	0.000
20191104T0011	0.912	0.897	0.887	0.874	0.844	0.788	0.706	0.391	0.083	0.000
20191101T2104	0.901	0.901	0.896	0.865	0.849	0.750	0.582	0.340	0.072	0.016
20191103T0520	0.913	0.904	0.876	0.868	0.844	0.778	0.652	0.443	0.129	0.000
20190830T0714	0.793	0.751	0.734	0.691	0.585	0.492	0.328	0.174	0.066	0.000
20190830T1443	0.732	0.724	0.702	0.630	0.592	0.531	0.379	0.199	0.075	0.000
20190911T1922	0.765	0.721	0.684	0.643	0.557	0.508	0.365	0.174	0.067	0.000
20190830T2019	0.858	0.830	0.782	0.762	0.687	0.609	0.446	0.278	0.082	0.000
20190829T1458	0.774	0.748	0.724	0.709	0.633	0.555	0.423	0.265	0.092	0.000
20190829T2020	0.739	0.670	0.642	0.621	0.573	0.513	0.402	0.180	0.059	0.000
20190830T0145	0.843	0.815	0.778	0.739	0.711	0.586	0.442	0.245	0.052	0.000
20190902T0946	0.914	0.907	0.890	0.864	0.852	0.793	0.692	0.497	0.150	0.000
20190905T1826	0.915	0.909	0.892	0.869	0.833	0.762	0.636	0.418	0.124	0.016
20190904T2004	0.896	0.878	0.873	0.838	0.791	0.665	0.489	0.235	0.040	0.000
20190907T1545	0.850	0.809	0.784	0.747	0.714	0.612	0.446	0.229	0.058	0.000
20190908T0352	0.836	0.790	0.731	0.710	0.678	0.566	0.415	0.259	0.069	0.000
20190906T0851	0.856	0.844	0.835	0.805	0.724	0.640	0.508	0.277	0.058	0.000
20191102T0140	0.916	0.911	0.896	0.880	0.858	0.813	0.685	0.442	0.190	0.000
20191103T0959	0.929	0.929	0.921	0.875	0.846	0.766	0.641	0.437	0.117	0.000
20190907T0934	0.921	0.908	0.902	0.887	0.848	0.796	0.663	0.457	0.130	0.000
20191104T0450	0.915	0.894	0.886	0.877	0.846	0.782	0.656	0.394	0.118	0.000
20191102T0615	0.902	0.891	0.886	0.873	0.819	0.745	0.661	0.410	0.158	0.000

Table A mAP@IOU[0.5:0.95] scores for each Mask R-CNN model trained in the Zanzibar dataset grid search. See Section 3.4.3 for model hyperparameters.

B Data Collection Camera Settings

Setting	Value
ISO	Auto
Mode	Time Value (shutter priority)
Shutter Speed	1/1000 in high light 1/800 in low light
Exposure Compensation	0.5-1 depending on light level
Lense	Autofocus
Camera Autofocus	Spot Focus

Table B Camera settings during data collection as outlined in Chapter 4. Any settings not displayed remained at default.

C NDD20 Above Water Example Images Class Labels

Filename	Labels		
	object	species	ID
911	dolphin	BND	-
421	dolphin	BND	-
241	dolphin	BND	-
	dolphin	BND	-
	dolphin	BND	-
177	dolphin	BND	30
296	dolphin	BND	7
1547	dolphin	BND	-
2078	dolphin	BND	-
	dolphin	BND	-
549	dolphin	BND	-
1891	dolphin	BND	-
1055	dolphin	BND	-
	dolphin	BND	-
797	dolphin	BND	-
61	dolphin	BND	-
1137	dolphin	BND	-
365	dolphin	WBD	-
1010	dolphin	BND	2
	dolphin	BND	-
450	dolphin	BND	-
	dolphin	BND	-
	dolphin	BND	-

Table C Class labels for the images shown in Figure 4.2. Table ordering is consistent with that of the Figure, viewing left to right.

D NDD20 Below Water Example Images Class Labels

Filename	Labels		
	object	out of focus	ID
26	dolphin	false	71
	dolphin	true	-
2116	dolphin	false	18
803	dolphin	false	11
1554	dolphin	false	49
1857	dolphin	false	41
	dolphin	false	-
1868	dolphin	false	30
	dolphin	true	-
	dolphin	true	-
509	dolphin	false	74
1065	dolphin	true	-
	dolphin	false	47
	dolphin	true	-
1136	dolphin	false	2
383	dolphin	false	7
311	dolphin	false	30
1602	dolphin	true	-
	dolphin	false	12
441	dolphin	true	-
	dolphin	true	-
	dolphin	false	50
522	dolphin	false	51
	dolphin	false	76
1341	dolphin	false	72
	dolphin	true	-
	dolphin	true	-
2116	dolphin	false	47

Table D Class labels for the images shown in Figure 4.3. Table ordering is consistent with that of the Figure, viewing left to right.

Bibliography

[1] (2009). Marine and Coastal Access Act.

[2] 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.

[3] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. Number: arXiv:1907.10902 arXiv:1907.10902 [cs, stat].

[4] Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6.

[5] 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.

[6] Alves, F., Quérouil, S., Dinis, A., Nicolau, C., Ribeiro, C., Freitas, L., Kaufmann, M., and Fortuna, C. (2013). Population structure of short-finned pilot whales in the oceanic archipelago of Madeira based on photo-identification and genetic analyses: implications for conservation. *Aquatic Conservation: Marine and Freshwater Ecosystems*, 23(5):758–776.

[7] 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.

[8] Araujo, G., Agustines, A., Tracey, B., Snow, S., Labaja, J., and Ponzo, A. (2019). Photo-ID and telemetry highlight a global whale shark hotspot in Palawan, Philippines. *Scientific Reports*, 9(1):17209.

[9] Araujo, G., Snow, S., So, C. L., Labaja, J., Murray, R., Colucci, A., and Ponzo, A. (2017). Population structure, residency patterns and movements of whale sharks in Southern Leyte, Philippines: results from dedicated photo-ID and citizen science. *Aquatic Conservation: Marine and Freshwater Ecosystems*, 27(1):237–252.

[10] Araújo, V. M., Britto Jr., A. S., Oliveira, L. S., and Koerich, A. L. (2022). Two-view fine-grained classification of plant species. *Neurocomputing*, 467:427–441.

[11] Armstrong, A. O., Armstrong, A. J., Bennett, M. B., Richardson, A. J., Townsend, K. A., and Dudgeon, C. L. (2019). Photographic identification and citizen science combine to reveal long distance movements of individual reef manta rays Mobula alfredi along Australia's east coast. *Marine Biodiversity Records*, 12(1):14.

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

[13] Atapour-Abarghouei, A., Bonner, S., and McGough, A. S. (2019). A Kings Ransom for Encryption: Ransomware Classification using Augmented One-Shot Learning and Bayesian Approximation. *arXiv:1908.06750 [cs]*. arXiv: 1908.06750.

[14] 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.

[15] 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.

[16] Barrowclift, E., Temple, A., Stead, S., Jiddawi, N., and Berggren, P. (2017). Social, economic and trade characteristics of the elasmobranch fishery on Unguja Island, Zanzibar, East Africa. *Marine Policy*, 83:128–136.

[17] 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.

[18] Bedetti, A., Greyling, C., Paul, B., Blondeau, J., Clark, A., Malin, H., Horne, J., Makukule, R., Wilmot, J., Eggeling, T., Kern, J., and Henley, M. (2020). System for Elephant Ear-pattern Knowledge (SEEK) to identify individual African elephants. *Pachyderm*, 61:63–77.

[19] Beery, S. and Bondi, E. (2021). Can poachers find animals from public camera trap images? *arXiv:2106.11236 [cs, q-bio]*. arXiv: 2106.11236.

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

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

[22] 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.

[23] 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.

[24] Berggren, P., Amir, O. A., Guissamulo, A., Jiddawi, N. S., Ngazy, Z., Stensland, E., Särnblad, A., and Cockroft, V. G. (2007). Sustainable Dolphin Tourism in East Africa. *WIOMSA Book Series*. Publisher: Newcastle University.

[25] Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. page 9.

[26] Bessesen, B., Oviedo, L., Burdett Hart, L., Herra-Miranda, D., Pacheco-Polanco, J., Baker, L., Saborío-Rodriguez, G., Bermúdez-Villapol, L., and Acevedo-Gutiérrez, A. (2014). Lacaziosis-like disease among bottlenose dolphins Tursiops truncatus photographed in Golfo Dulce, Costa Rica. *Diseases of Aquatic Organisms*, 107(3):173–180.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] Boulais, O., Alaba, S. Y., Ball, J. E., Campbell, M., Iftekhar, A. T., Moorehead, R., and Primrose, J. (2021). SEAMAPD21: a large-scale reef sh dataset for ne-grained categorization. page 7.

[34] Bouma, S., Pawley, M. D., Hupman, K., and Gilman, A. (2018). 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.

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

[36] Brust, C.-A., Burghardt, T., Groenenberg, M., Kading, C., Kühl, H. S., Manguette, M. L., and Denzler, J. (2017). Towards Automated Visual Monitoring of Individual Gorillas in the Wild. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2820–2830. ISSN: 2473-9944.

[37] 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.

[38] 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.

[39] 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.

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

[41] 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.

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

[43] 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.

[44] 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.

[45] Cheney, B., Corkrey, R., Durban, J. W., Grellier, K., Hammond, P. S., Islas-Villanueva, V., Janik, V. M., Lusseau, S. M., Parsons, K. M., Quick, N. J., Wilson, B., and Thompson, P. M. (2014). Long-term trends in the use of a protected area by small cetaceans in relation to changes in population status. *Global Ecology and Conservation*, 2:118–128.

[46] 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.

[47] 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).

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

[49] Christiansen, F., Lusseau, D., Stensland, E., and Berggren, P. (2010). Effects of tourist boats on the behaviour of Indo-Pacific bottlenose dolphins off the south coast of Zanzibar. *Endangered Species Research*, 11:91–99.

[50] 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.

[51] Clapham, M., Miller, E., Nguyen, M., and Darimont, C. T. (2020). Automated facial recognition for wildlife that lack unique markings: A deep learning approach for brown bears. *Ecology and Evolution*, n/a(n/a).

[52] 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.

[53] 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.

[54] 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.

[55] Currie, J. J., Stack, S. H., and Kaufman, G. D. (2018). Conservation and Education Through Ecotourism: Using Citizen Science to Monitor Cetaceans in the Four-Island Region of Maui, Hawaii. *Tourism in Marine Environments*, 13(2):65–71.

[56] 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.

[57] 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.

[58] 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.

[59] Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Llados, J., and Pal, U. (2017). SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification. Number: arXiv:1707.02131 arXiv:1707.02131 [cs].

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

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

[62] 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.

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

[64] Evans, P. G. H. and Hammond, P. S. (2004). Monitoring cetaceans in European waters. *Mammal Review*, 34(1-2):131–156.

[65] 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.

[66] 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.

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

[68] 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.

[69] 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.

[70] Franklin, T., Smith, F., Gibbs, N., Childerhouse, S., Paton, D., Franklin, W., Baker, S., and Clapham, P. (2008). Migratory movements of humpback whales (Megaptera novaeangliae) between eastern Australia and the Balleny Islands, Antarctica, confirmed by photo-identification. page 5.

[71] 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.

[72] Gal, Y. (2016). Uncertainty in Deep Learning. page 174.

[73] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. Number: arXiv:1506.02142 arXiv:1506.02142 [cs, stat].

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

[75] Gao, J., Burghardt, T., Andrew, W., Dowsey, A. W., and Campbell, N. W. (2021). Towards Self-Supervision for Video Identification of Individual Holstein-Friesian Cattle: The Cows2021 Dataset. *arXiv:2105.01938 [cs]*. arXiv: 2105.01938.

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

[77] Gholamalinezhad, H. and Khosravi, H. (2020). Pooling Methods in Deep Neural Networks, a Review. page 16.

[78] 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.

[79] 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).

[80] 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.

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

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

[83] Goswami, V. R., Madhusudan, M. D., and Karanth, K. U. (2007). Application of photographic capture–recapture modelling to estimate demographic parameters for male Asian elephants. *Animal Conservation*, 10(3):391–399.

[84] 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.

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

[86] Gómez, R. (2019). Understanding Ranking Loss, Contrastive Loss, Margin Loss, Triplet Loss, Hinge Loss and all those confusing names.

[87] 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.

[88] 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.

[89] 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.

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

[91] Harislqbal88 (2018). PlotNeuralNet.

[92] 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.

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

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

[95] 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.

[96] 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.

[97] Hermans, A., Beyer, L., and Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. arXiv:1703.07737 [cs].

[98] 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.

[99] Hoffer, E. and Ailon, N. (2018). Deep metric learning using Triplet network. *arXiv:1412.6622 [cs, stat]*. arXiv: 1412.6622.

[100] 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.

[101] 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.

[102] Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. (2018). Variational Bayesian dropout: pitfalls and fixes. Number: arXiv:1807.01969 arXiv:1807.01969 [cs, stat].

[103] 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.

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

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

[106] 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.

[107] Karpathy, A. (2017). A peek at trends in machine learning.

[108] Kay, J. and Merrifield, M. (2021). The Fishnet Open Images Database: A Dataset for Fish Detection and Fine-Grained Categorization in Fisheries. *arXiv:2106.09178 [cs]*. arXiv: 2106.09178.

[109] Keen, E. M., Wren, J., O'Mahony, , and Wray, J. (2021). catRlog: a photo-identification project management system based in R. *Mammalian Biology*.

[110] Kendall, A. and Gal, Y. (2017). What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? Number: arXiv:1703.04977 arXiv:1703.04977 [cs].

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

[112] Kim, P. (2017). Convolutional Neural Network. In *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*, pages 121–147. Apress, Berkeley, CA.

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

[114] Korotcov, A., Tkachenko, V., Russo, D. P., and Ekins, S. (2017). Comparison of Deep Learning With Multiple Machine Learning Methods and Metrics Using Diverse Drug Discovery Data Sets. *Molecular Pharmaceutics*, 14(12):4462–4475.

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

[116] 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.

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

[118] Kulits, P., Wall, J., Bedetti, A., Henley, M., and Beery, S. (2021). ElephantBook: A Semi-Automated Human-in-the-Loop System for Elephant Re-Identification. *arXiv:2106.15083 [cs]*. arXiv: 2106.15083.

[119] 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.

[120] Laptev, N., Yosinski, J., Li, L. E., and Smyl, S. (2017). Time-series Extreme Event Forecasting with Neural Networks at Uber. page 5.

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

[122] 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.

[123] 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.

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

[125] 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.

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

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

[128] 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.

[129] 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.

[130] 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.

[131] 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.

[132] 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.

[133] 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.

[134] 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.

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

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

[137] 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.

[138] Luo, Z., Liu, H., and Wu, X. (2005). Artificial neural network computation on graphic process unit. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 1, pages 622–626 vol. 1. ISSN: 2161-4407.

[139] 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.

[140] Maji, S. (2011). Large Scale Image Annotations on Amazon Mechanical Turk. page 12.

[141] 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.

[142] Mann, J. (1999). Behavioral Sampling Methods for Cetaceans: A Review and Critique. *Marine Mammal Science*, 15(1):102–122.

[143] Mann, J. (2019). Dolphin Matching Using Google Cloud Vision (DMUGCV), available at https://drive.google.com/file/d/1nLXJiSjGCgvjW54UTd46EU64oJcKS1f_/.

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

[145] 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.

[146] 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.

[147] 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.

[148] Moindrot, O. (2018). Triplet Loss and Online Triplet Mining in TensorFlow.

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

[150] 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.

[151] 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.

[152] 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.

[153] 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.

[154] 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.

[155] 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.

[156] 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.

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

[158] 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.

[159] Osband, I. (2016). Risk versus Uncertainty in Deep Learning: Bayes, Bootstrap and the Dangers of Dropout. page 5.

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

[161] 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.

[162] Paterakis, N. G., Mocanu, E., Gibescu, M., Stappers, B., and van Alst, W. (2017). Deep learning versus traditional machine learning methods for aggregated energy demand prediction. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6.

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

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

[165] 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.

[166] 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.

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

[168] 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.

[169] 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.

[170] 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.

[171] 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.

[172] 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.

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

[174] 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.

[175] 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.

[176] 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.

[177] 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.

[178] 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.

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

[180] 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.

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

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

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

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

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

[186] Shane, S. H., Wells, R. S., and Wursig, B. (1986). ECOLOGY, BEHAVIOR AND SOCIAL ORGANIZATION OF THE BOTTLENOSE DOLPHIN: A REVIEW. *MARINE MAMMAL SCIENCE*, 2(1):30.

[187] 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/.

[188] Sharpe, M. and Berggren, P. (2019a). 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.

[189] Sharpe, M. and Berggren, P. (2019b). 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.

[190] Shirke, A., Saifuddin, A., Luthra, A., Li, J., Williams, T., Hu, X., Kotnana, A., Kocabalkanli, O., Ahuja, N., Green-Miller, A., Condotta, I., Dilger, R. N., and Caesar, M. (2021). Tracking Grow-Finish Pigs Across Large Pens Using Multiple Cameras. *arXiv:2111.10971 [cs]*. arXiv: 2111.10971.

[191] Silva, M. A., Steiner, L., Cascão, I., Cruz, M. J., Prieto, R., Cole, T., Hamilton, P. K., and Baumgartner, M. (2012). Winter sighting of a known western North Atlantic right whale in the Azores. page 6.

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

[193] Smolker, R. A., Richards, A. F., Connor, R. C., and Pepper, J. W. (1992). Sex Differences in Patterns of Association Among Indian Ocean Bottlenose Dolphins. *Behaviour*, 123(1-2):38–69. Publisher: Brill.

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

[195] Soviany, P. and Ionescu, R. T. (2018). Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction. *arXiv:1803.08707 [cs]*. arXiv: 1803.08707.

[196] 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.

[197] Stephenson, F., Polunin, N. V. C., Mill, A. C., Scott, C., Lightfoot, P., and Fitzsimmons, C. (2017). Spatial and temporal changes in pot-fishing effort and habitat use. *ICES Journal of Marine Science*, 74(8):2201–2212.

[198] 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.

[199] 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.

[200] 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.

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

[202] 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.

[203] Temple, A. J., Kiszka, J. J., Stead, S. M., Wambiji, N., Brito, A., Poonian, C. N. S., Amir, O. A., Jiddawi, N., Fennessy, S. T., Pérez-Jorge, S., and Berggren, P. (2018). Marine megafauna interactions with small-scale fisheries in the southwestern Indian Ocean: a review of status and challenges for research and management. *Reviews in Fish Biology and Fisheries*, 28(1):89–115.

[204] Temple, A. J., Stead, S. M., Jiddawi, N., Wambiji, N., Dulvy, N. K., Barrowclift, E., and Berggren, P. (2020). Life-history, exploitation and extinction risk of the data-poor Baraka's whipray (Maculabatis ambigua) in small-scale tropical fisheries. *Journal of Fish Biology*, 97(3):708–719.

[205] Temple, A. J., Wambiji, N., Poonian, C. N., Jiddawi, N., Stead, S. M., Kiszka, J. J., and Berggren, P. (2019). Marine megafauna catch in southwestern Indian Ocean small-scale fisheries from landings data. *Biological Conservation*, 230:113–121.

[206] Temple, A. J., Westmerland, E., and Berggren, P. (2021). By-catch risk for toothed whales in global small-scale fisheries. *Fish and Fisheries*, 22(6):1155–1159.

[207] Thompson, J. W., Zero, V. H., Schwacke, L. H., Speakman, T. R., Quigley, B. M., Morey, J. S., and McDonald, T. L. (2022). finFindR: Automated recognition and identification of marine mammal dorsal fins using residual convolutional neural networks. *Marine Mammal Science*, 38(1):139–150.

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

[209] 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.

[210] Timm, M., Maji, S., and Fuller, T. (2018). Large-Scale Ecological Analyses of Animals in the Wild Using Computer Vision. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1977–19772, Salt Lake City, UT. IEEE.

[211] 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.

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

[213] 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.

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

[215] 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.

[216] 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.

[217] 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.

[218] 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*.

[219] 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.

[220] Vetrova, V., Coup, S., Frank, E., and Cree, M. J. (2018). Hidden Features: Experiments with Feature Transfer for Fine-Grained Multi-Class and One-Class Image Categorization. In *2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, Auckland, New Zealand. IEEE.

[221] Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018:e7068349. Publisher: Hindawi.

[222] 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.

[223] Véronique, S. (2022). Underwater photo-identification of sperm whales (Physeter macrocephalus) off Mauritius. page 17.

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

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

[226] Wang, Q., Min, W., He, D., Zou, S., Huang, T., Zhang, Y., and Liu, R. (2020a). Discriminative fine-grained network for vehicle re-identification using two-stage re-ranking. *Science China Information Sciences*, 63(11):212102.

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

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

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

[230] 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.

[231] Weigmann, S., Gon, O., Leeney, R. H., Barrowclift, E., Berggren, P., Jiddawi, N., and Temple, A. J. (2020). Revision of the sixgill sawsharks, genus Pliotrema (Chondrichthyes, Pristiophoriformes), with descriptions of two new species and a redescription of P. warreni Regan. *PLOS ONE*, 15(3):e0228791.

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

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

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

[235] 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.

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

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

[238] 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.

[239] 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.

[240] 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.

[241] 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.

[242] Yang, L., Luo, P., Loy, C. C., and Tang, X. (2015). A large-scale car dataset for fine-grained categorization and verification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3973–3981, Boston, MA, USA. IEEE.

[243] Yang, L., Sharpe, M., Temple, A. J., and Berggren, P. (2021). Characterization and comparison of echolocation clicks of white-beaked dolphins (Lagenorhynchus albirostris) off the Northumberland coast, UK. *The Journal of the Acoustical Society of America*, 149(3):1498–1506.

[244] Yang, L., Sharpe, M., Temple, A. J., Jiddawi, N., Xu, X., and Berggren, P. (2020). Description and classification of echolocation clicks of Indian Ocean humpback (Sousa plumbea) and Indo-Pacific bottlenose (Tursiops aduncus) dolphins from Menai Bay, Zanzibar, East Africa. *PLOS ONE*, 15(3):e0230319.

[245] Yang, L., Xu, X., and Berggren, P. (2022). Influence of ice concentration and thickness on under-ice ambient noise levels in shallow coastal waters of Liaodong Bay, China. *Estuarine, Coastal and Shelf Science*, 266:107706.

[246] Yang, L., Xu, X., Zhang, P., Han, J., Li, B., and Berggren, P. (2017). Classification of underwater vocalizations of wild spotted seals (Phoca largha) in Liaodong Bay, China. *The Journal of the Acoustical Society of America*, 141(3):2256–2262.

[247] 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.

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

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

[250] Zhang, L., Tan, J., Han, D., and Zhu, H. (2017b). From machine learning to deep learning: progress in machine intelligence for rational drug discovery. *Drug Discovery Today*, 22(11):1680–1685.

[251] 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.

[252] 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.

[253] 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.

[254] Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2017). Random Erasing Data Augmentation. *arXiv:1708.04896 [cs]*. arXiv: 1708.04896.

[255] 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.