# Photo-Identification of Marine Cetaceans Using Convolutional Neural Networks

**Cameron Trotter**

School of Electrical and Electronic Engineering

Newcastle University

In Partial Fulfilment of the Requirements for the Degree of

*Doctor of Philosophy*

SUBMISSION DATE

DEDICATION

# Declaration

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

Cameron Trotter
SUBMISSION DATE

# Acknowledgements

ACKNOWLEDGEMENTS

# Abstract

ABSTRACT

# Table of contents

# List of figures

# List of tables

# Chapter 1

# Introduction

Modelling cetacean (whale, dolphin, and porpoise) population dynamics and behaviour is paramount to effective population management and conservation. Robust data is required for the design and implementation of conservation strategies and to assess the risks presented by anthropogenic activity such as offshore wind turbines and commercial fishing. Moreover, cetaceans make prime candidates for modelling ecosystem change under the ecosystem sentinel concept as they reflect the current state of the ecosystem and respond to change across different spatial and temporal scales [66]. 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 [21, 105]. 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.

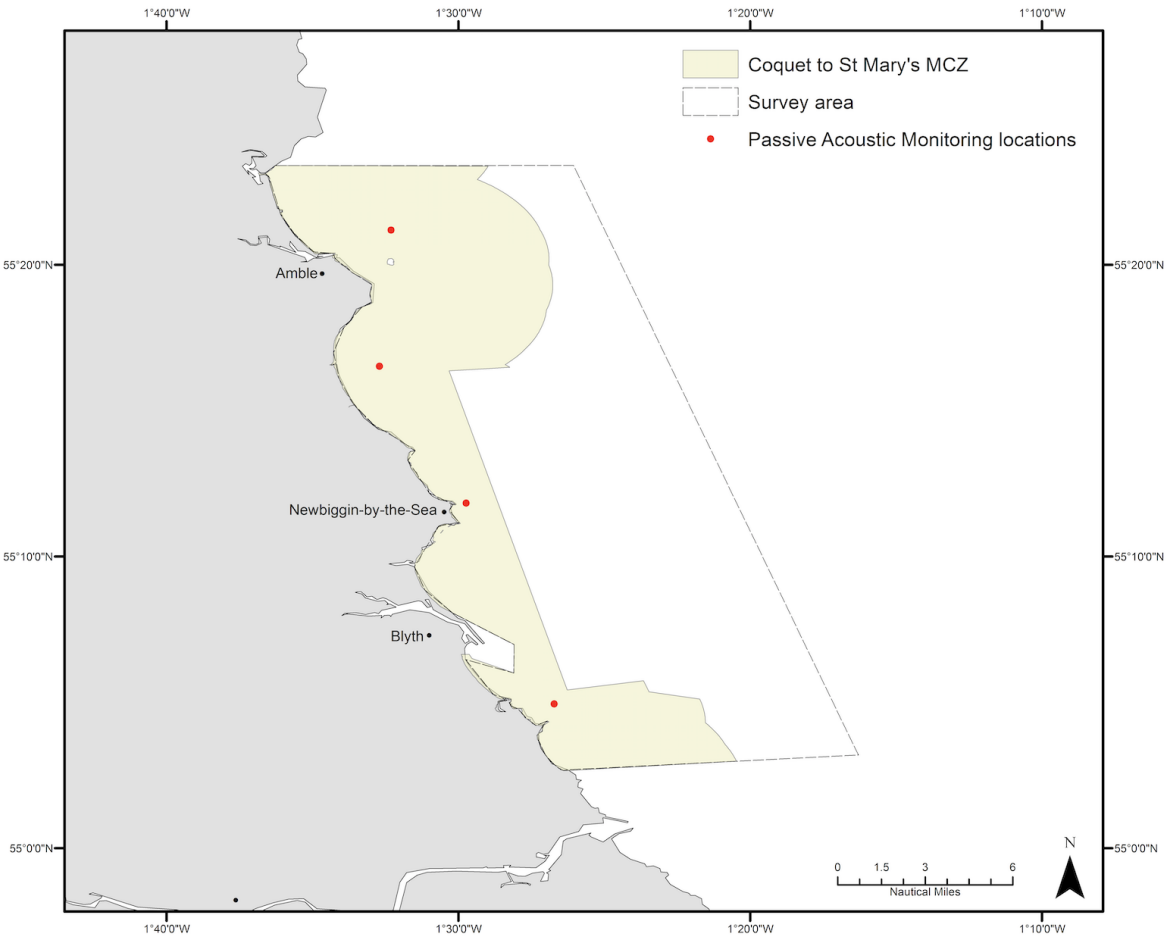


Figure 1.1 Map of the survey area, Northumberland, UK, from St. Mary's Lighthouse in the south to 25nm above Coquet Island in the north.

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 [29, 36] 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 [96]. 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

## 2.1 Introduction

In recent years, deep learning has become somewhat of a buzzword within the world of computing, but not without good reason. Deep learning models, those comprised of a large number of processing layers in order to learn multiple abstracted representations of the data provided, have consistently been shown to outperform other machine learning techniques, especially in the field of computer vision [51]. As the field of deep learning increases in scope every day, this Chapter will focus primarily on the world of deep learning in a computer vision context. A brief introduction to deep learning will be provided, before looking at how these methodologies have been applied to computer vision, specifically Convolutional Neural Networks (CNNs) and their use in object detection. Literature focusing on object detection in a marine cetacean space will be explored, as well as the current space of fine-grained computer vision.

## 2.2 A Brief Introduction to Deep Learning

Deep learning, a subfield of machine learning, aims to create artificial networks to complete tasks, i.e. learn, in a similar way to how neurons in the human brain operate. These computational models are often multiple neurons deep, known as layers, with lower layers representing basic abstractions, building up from this as you go *deeper* into the network, resulting in final layers of neurons which, based on information passed to them from lower levels, can begin to provide estimations of answers to a given problem. It is in this way that deep learning methods differ from older machine learning techniques, which were far more constrained in how data could be represented. Considerable domain expertise was required

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 be able to detect or classify the input.

Deep learning models in contrast are capable of performing 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 may be optimised to learn lines and basic shapes, middle layers may be optimised to learn more complex ideas such as how these lines and shapes fit together, with the final layers providing an output of object label (e.g. `dolphin`). It should be stressed however that the features these layers are looking for are not specified by humans, but rather learned from the data by updating their weights and biases, predominantly through stochastic gradient descent and backpropagation [42]. This is very similar to how the brain learns from the multi-modal data it receives from the body, capturing the intricacies of massive amounts of data using a connection of smaller optimised neurons.

This ambition to create networks similar to how the brain operates stems mainly from work undertaken in 1943 by McCulloch and Pitts [65] in an attempt to understand how neurons in the brain allow for the understanding of complex patters. This model formed the basis of future work into machine learning, and thus deep learning. This work continued at small scale for many years. It has only been recently thanks to advances in availability and power of the computing resources available has deep learning research accelerated. The transition away from model training on CPUs to multiple high-powered GPUs has been one of the largest advancements in deep learning, allowing for a significant speed-up in the train time for these models, resulting in much more prototyping in a smaller time frame. Further to this, the advent of cloud computing has allowed for much more cost-effective model development. Thanks to the Pay As You Go model of computing now commonplace, cloud providers such as Microsoft, Amazon, and Google have eliminated the need for researchers to procure their own hardware required for training, which can in some cases be prohibitively expensive.

More so, advances in deep learning have been helped greatly through the development of standard development frameworks. Google's Tensorflow [1] and Facebook's PyTorch [72] allow for researchers to develop models much faster than previously due to their reduction in the amount of boiler-plate code needed, with these frameworks often doing a lot of the *heavy lifting* in the background. Further advances have been made through the availability of high quality coarse-grained datasets such as MNIST [52], Caltech-256 [34], and ImageNet [23] 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 [71]. 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 [88], batch normalisation [45], stochastic gradient descent with warm restarts [63], mixup [110], as well as various forms of data augmentation.

### 2.2.1 Supervised and Unsupervised Learning

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

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

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

**Unsupervised learning** tasks are, in contrast, those where prior ground truths for your data are not available. As such, no classification can be performed using this method, however it is often used for clustering and aiding in understanding of the underlying data structure. These unsupervised algorithms, such as K-Means clustering [38], are not given any guidance on how to define the data into clusters but are rather left to discover interesting structure on their own.

Taking our dog and cat data again as an example, it is clear how this data could be clustered in an unsupervised manner. Asking the clustering algorithm to provide two clusters for the data (e.g. $K = 2$) , a model could be trained to split the data with all dogs in one

cluster and all cats in the other, without having to be told which images are dogs and which are cats.

### 2.2.2 The Stochastic Gradient Descent Algorithm

In order to generalise our deep learning models, we need to be able to optimise the weights and biases within each individual neuron. Most commonly, this is performed using gradient descent to minimise a loss function (a measure of distance between ground truth and prediction) in such a way that weights are updated in the *opposite* direction of the gradient of the loss function. As such, we follow the direction of the slope of the surface created by the loss function downhill iteratively until we reach a minima, an area where the loss is lowest [83]. Before the advent of deep learning and big data, it was common for the whole dataset to be used to compute the gradient at each iteration; however due to the size of modern day datasets this is no longer possible due to the computational cost this would impose on the system. As such, batches of the dataset are often used to give an estimation of the overall loss gradient.

In order to reach this minimum, a process known as stochastic gradient descent (SGD) is most commonly used. At each iteration of SGD, a batch will only contain one randomly selected training example. The loss for this example is calculated, and used to step down the gradient slope, rather than the sum of the loss' gradient over all training examples. As we only take one example per iteration, the path taken down the slope to the minima is far noisier and random than the path obtained from using all examples, hence the *stochastic* nature of the gradient descent. This stochastic nature does result in a longer convergence time to the minima compared to a regular gradient descent, however this is outweighed by the reduction in computational expense. The use of SGD often leads to a good set of model weights quickly compared to other, more elaborate techniques [8].

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 [63], Momentum [74], RMSProp [93], Adam [49], and AMSGrad [77]. 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 [19, 22].

### 2.2.3 Backpropagation

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

Backpropagation was originally described in Linnainmaa's masters thesis [58], although it's effect was not fully realised until 1986, when Rumelhart *et al.* discussed the advantages to using backpropagation over other learning approaches [84]. Whilst multiple attempts have been made to improve the original algorithm [5, 53, 56, 57, 70], these are rarely adopted by deep learning researchers as little improvement can be gained from them compared to the overhead of modifying existing deep learning frameworks to incorporate the changes.

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

## 2.3 Deep Learning for Computer Vision

The field of computer vision, allowing computers to gain and interpret knowledge from image and video data, is one area where deep learning has excelled. Generalisable concepts such as CNNs have quickly become commonplace for solving computer vision tasks, in most cases replacing the need for hand-crafted pipelines specialised to the task at hand, thanks to their ability to learn complex patterns in data where there is a strong spatial and temporal dependency between the values. This ability is essential for processing image data which is, at it's most basic level, a matrix of pixel values. These matrices are three dimensional, representing an image's height, width, and depth, where depth is dependant on the colour model used to represent the image. The most common of these models is RGB which has

channels representing the red, green, and blue visual light present in an image; as such a matrix representing an RGB image with have a depth of three. Other colour models have varying depth values, a greyscale image would have a depth of one for example, whilst a CMYK image would have a depth of four. As can be imagined, these matrices can very quickly reach massive sizes. An RGB image at 1080p resolution for example would require a matrix of size 1920x1080x3, or 6220800 values. CNNs allow us to reduce the image down to a more workable form whilst still retaining key features which will allow us to infer predictions.

### 2.3.1 Convolutional Neural Networks

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

#### Convolutional Layers

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

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

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

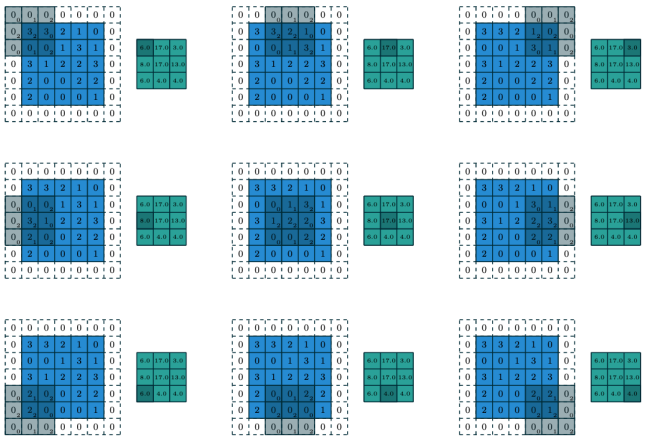

Figure 2.1 A visual representation of convolution. A kernel, represented by the grey squares, operates over a padded input matrix, blue, with a stride of 2 to produce an output feature map, green. Note that the kernel's weights are denoted by the number in the lower right of each box and the input pixel value is denoted by the number in the middle of each box. [26]

map of equal size to the input, whereas a stride of 2 would skip every other pixel, reducing the output feature map by half.

A problem can arise during convolution when the kernel reaches the edges of the input matrix. As there is nothing past the edge values, these values must be trimmed as they are never in the centre of the kernel. This can cause the reduction of the matrix, as some values are never utilised, which may be detrimental when we require an output feature map which is the same size as that of the input. As such, padding can be performed. This technique places zeros around the edges of the input matrix, expanding it's size and allowing pixel values formerly at the edges to be utilised by the kernel. A visual representation of how convolution is performed can be seen in Figure 2.1.

### Pooling Layers

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

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

A number of different pooling layer architectures exist in literature, such as max pooling, average pooling [11], and stochastic pooling [109], although more recent architectures favour the use of strided convolutions, rather than pooling, to reduce the size of the output feature map.

### Fully Connected Layers

Fully connected layers take feature maps produced by the preceding convolutional and pooling layers and reduce these down to a single $N$-dimensional vector, where $N$ represents the total number of classes and each dimension's value is the probability of the class. These probabilities are achieved using a softmax activation function, which takes the exponents of each input and normalises them by the sum of all inputs, giving values between 0 and 1. Outputted $N$-dimensional vectors can then be considered a feature map in their own right for further processing [50] or as a category for classification as the last layer of the network [33].

### Layer Architectures

Using the three layer types described above it is possible to create, in theory, an infinite number of CNN architectures all with different amounts and combinations of layers. There is no guarantee that every possible architecture will perform well however (indeed, one possible combination would be a single fully connected layer, which would not perform well at all). 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 [27], 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 [52] 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 [50]. Utilising a similar but deeper architecture to LeNet, with more filters and a larger number of stacked convolutional layers, AlexNet also included now common CNN building blocks such as dropout [88], max pooling [11], and ReLU activation functions; the most popular non-linear activation function currently in deep learning, especially in computer vision [41].

The activation function is responsible for deciding which neuron in the layer passes it's value to the layer below by computing the weighted sum of the inputs and passing the result through a non-linear function. ReLU's non-linear function returns 0 for any negative value, or

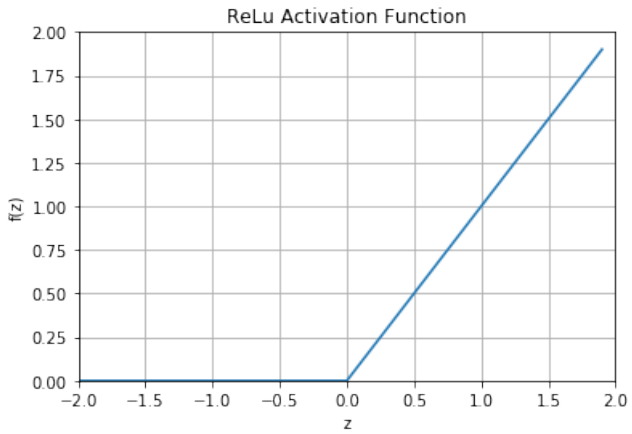


Figure 2.2 A visualisation of the ReLU Activation Function

for any positive value $x$, it returns $x$. This can be written as $f(x) = max(0,x)$, and visualised in Figure 2.2. 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 [89]. This net achieved a top-5 error rate of 6.67%, very close to what untrained humans could achieve on the competition dataset. This was achieved through a 22 layer deep CNN utilising several small convolutions, reducing the number of parameters from 60million in AlexNet to 4million in GoogleNet.

Finally ResNet was introduced a year later at ILSVRC15. This architecture can be as large as 152 layers deep, and achieved a human-beating top-5 error rate of 3.57% [40]. Shallower versions of ResNet exist, such as ResNet50 and ResNet101, which are 50 and 101 layers deep respectively.

### 2.3.2 Object Detection

As discussed, CNNs are now one of the main tools available for computer vision tasks. Object detection tasks are no exception, with CNNs now being utilised en masse. These tasks concern themselves with attempting to identify and segment distinct classes of objects found in images and video, and is often performed in one of two ways.

#### Region Proposal Networks

The first, known as a Region Proposal Network (RPN), attempts to find image regions likely to contain objects of given classes. Training data is usually provided in the form of bounding boxes drawn around objects of interest and labelled with the corresponding class. These

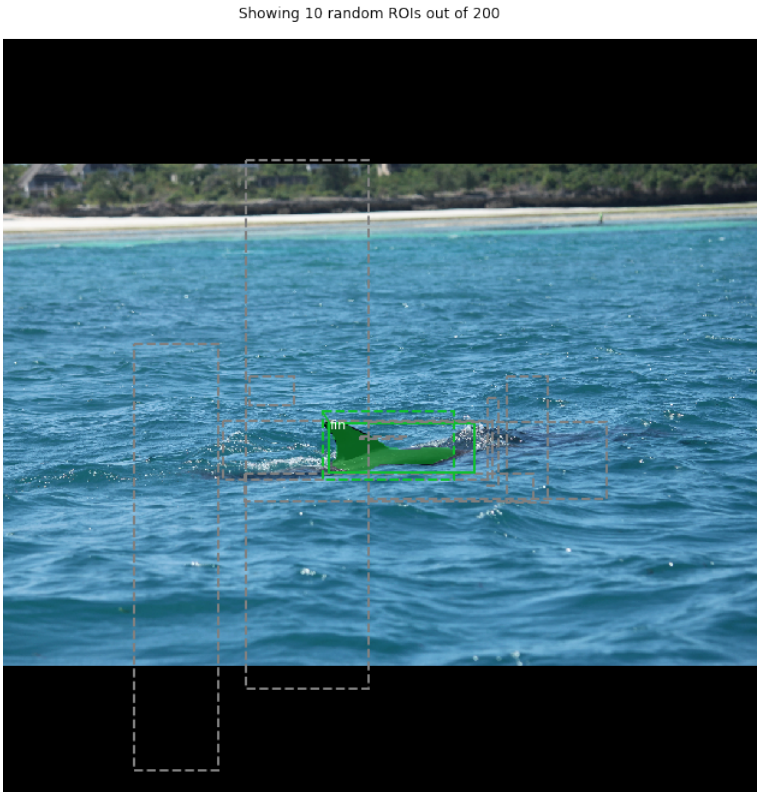


Figure 2.3 An example of ROIs generated on an image by an RPN, showing 10 random proposals out of the 200 generated. Note that two of the ROIs have been successfully classified as `fin`.

RPN detections can be relatively fast, using a selection search [94] gives around 2000 region proposals, known as ROIs, in only a few seconds on a CPU. Example ROIs can be seen in Figure 2.3.

Selection search is most commonly used with the R-CNN object detection algorithm [33]. This algorithm has a high recall rate due to the large amount of proposals, as there is a high probability that some of these proposals will contain Regions of Interest (ROIs) containing the objects being searched for. However, this can be time consuming and computationally expensive (although less computationally expensive than just sliding a window over the full image) as the network needs to be trained to classify these 2000 region proposals, taking up a large amount of disk space. Detection can also be slow using a vanilla R-CNN and, with the selection search being fixed, no adaptive learning takes place here which may lead to bad region proposals throughout.

Some of these time drawbacks were fixed in later versions of R-CNN, known as Fast-RCNN [32]. 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 dominated by how fast ROIs can be generated.

To fix this issue, Faster R-CNN was developed [80]. 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 [78] and SSD [61].

### 2.3.3 Semantic Segmentation

Along with object detection, semantic segmentation is one of the key research areas in computer vision. Rather than providing bounding boxes around objects of interest as output, semantic segmenters aim to provide fine-grain categorisation for every pixel in an image, grouping pixels together in object classes known as masks. An example of an image and it's masks can be seen in Figure 2.4.

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


Figure 2.4 An example of an image and it's corresponding ground truth `fin` masks.

a classification based on which highest scoring region it is contained in. Note that any pixels not in a region are given the class label of `background`. This is achieved through the use of a lightweight binary classifier ran over multiple proposal boxes, known as anchors, covering the image at different scales. Each anchor is given an object score denoted by Intersection Over Union (IOU), a measure of how much overlap there is between a model's predicted bounding box and the ground truth. This is taken at a set confidence threshold, usually 50%, as the model will predict potentially hundreds of boxes for an image, all with different confidence levels. The vast majority of these predictions will be wrong, but will also (hopefully) have very low confidence scores and so they can be safely ignored and thus not counted in evaluation metrics. Taking a predicted bounding box $B_p$ and a ground truth box $B_g$, the IOU between the two can be defined as:

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

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

Usually however, positive anchors do not fully cover the ground truth object. Because of this, the RPN regresses a refinement applied to the anchors, shifting and resizing them to correct their encasement of the ground truth object. An example of this can be seen in Figure 2.6.

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

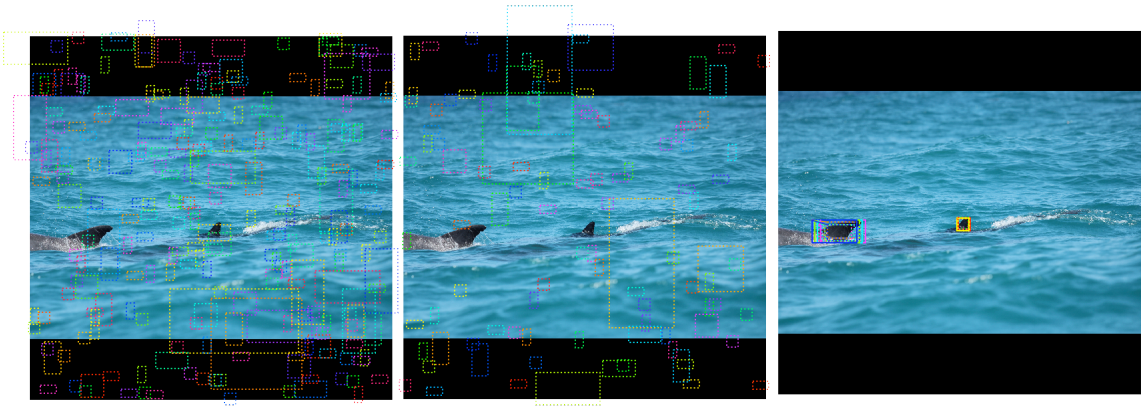

Figure 2.5 Generated anchors. Left: negative anchors. Middle: neutral anchors. Right: positive anchors.

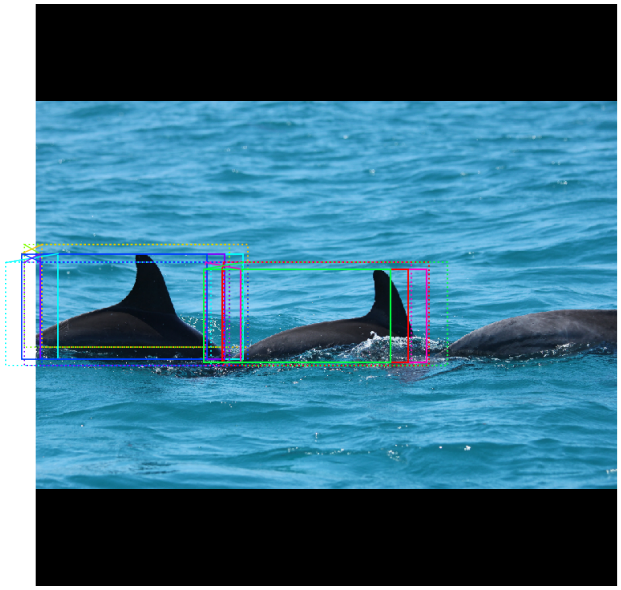

Figure 2.6 An example of refined anchors. Positive anchors before refinement are dotted, after refinement are solid.

do not contain enough feature information to generate well defined masks. Recent research has attempted to fix these issues, such as SDS [37] or Facebook's Mask R-CNN [39].

Second, a Fully Convolutional Network (FCN) can be utilised for semantic segmentation [62]. An FCN learns pixel to pixel mappings without the need for region proposals, whilst also only including convolutional and pooling layers, allowing for an input image of arbitrary size (compared to classical CNNs which are generally constrained by a preset image size). This does lead to the disadvantage of down sampling the resolution of the outputted feature maps, leading to sometimes ill-defined segmented boundaries. This issue has attempted to be corrected however with more advanced FCNs such as SegNets [4] and DeepLab [17].

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 [12].

### 2.3.4 Part Segmentation

Whilst fine-grained visual categorisation is still an area of new research, an emerging approach to tackling this problem is through the use of part segmentation, whereby a coarse-grained classification is broken down into sub-components which are then analysed to provide a fine-grained identification [111]. This is still an active area of research, with some approaches focusing on a form of hierarchical part matching [107], some on alignment of objects to define a super-class shape [30], some utilising deformable part descriptors [112], and others using part localisation [59].

### 2.3.5 Instance Segmentation

Building on the concept of semantic segmentation, instance segmentation can be performed when further detail about an image is required by a developed system. Whilst many of the underlying processes are similar between the two segmentation types, instance segmentation allows for the model to distinguish between multiple objects which are of the same class; an example of this can be seen in Figure 2.7.

As such, instance segmentation provides a far more detailed explanation of the input image. This information can be invaluable if your developed system is required not only to understand what pixel classes are present in the input image, but also how many of these class instances there are. In a sense, instance segmentation can be seen as combining both object detection and semantic segmentation into one task. Traditionally however, in order to

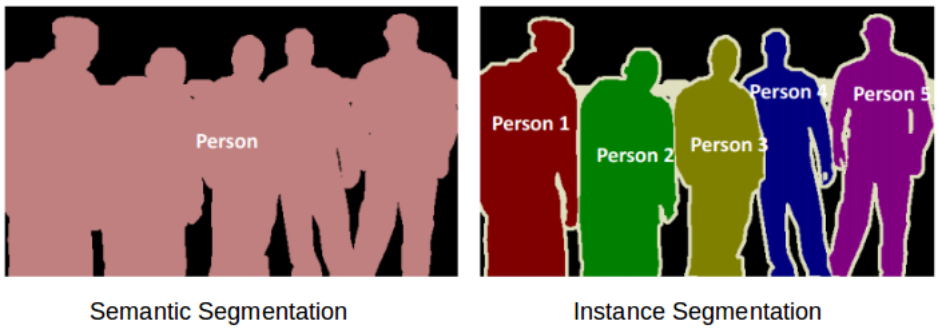


Figure 2.7 An example showing the difference between semantic and instance segmentation. In semantic segmentation, all pixels which belong to the `person` class have been classified as one `person` object. In instance segmentation, five `person` objects have been detected, and all `person` pixels have been assigned to one of the objects. Image from [86].

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 [80], and Mask R-CNN [39]. In contrast, *bottom-up* systems first segment then detect, such as SpatialEmbedding [67] which attempts to tackle instance segmentation through the use of a Gaussian function to produce a probability for a pixel being part of the background or foreground, and then performing object detection on the foreground pixels. *Bottom-up* approaches often fail to reach similar levels of performance as *top-down* approaches however, and are thus rarely used. The major similarity between both *top-down* and *bottom-up* approaches is that they are both sequential in nature, requiring one stage to happen before the other. As such, these systems are very hard to speed up and are far from real-time. However, two stage systems often perform the best in terms of accuracy, and thus are still extremely common backbones of systems requiring the use of instance segmentation.

In recent years research into the development of real-time instance segmentation has shifted to utilising a one stage approach. These one stage systems are often able to achieve near real-time performance, although often do not produce levels of segmentation accuracy seen when utilising two stage systems. ExtremeNet [114] 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 [108] utilises the concept of Chebyshev polynomials to fit a radius around each object inside of the detected bounding box. Similarly, PolarMask [106] also represents masks through the use of a contour around

the object, modelling this through the use of polar coordinates. FourierNet [81] 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 [7] builds on the well known YOLO object detection architecture, specifically YOLOv3 [79], 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 [16] works in a similar way to YOLOACT however predicts an attention map rather than instance coefficients and utilises FCOS [92] as a backbone, a completely anchor and proposal free object detection architecture resulting in reduced complexity when compared to YOLO [78] and SSD [61].

Whilst the majority of one stage approaches to instance segmentation rely on bounding boxes, this is not always the case. SOLO [99] introduces the concept of instance categories, assigning categories to each pixel according to the size and location of the instance. SOLOv2 [100] 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 [39]. Because of this, it is important

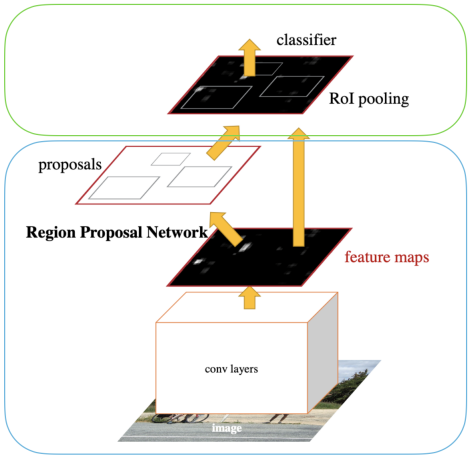


Figure 2.8 The Faster R-CNN architecture [80]. The blue box represents operations in stage one. The green box represents operations in stage two.

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.* [80] (itself an extension of Fast R-CNN developed in 2015 [32]).

Faster R-CNN is a two stage architecture. The first stage utilises a standard backbone network, typically ResNet [40], VGG [87], or Inception [89], to convert an input image into a set of feature maps which are passed to an RPN for analysis (see Section 2.3.2 for a breakdown of RPNs). This RPN generates region proposals which are passed to the second stage of Faster R-CNN, along with the previously generated feature maps, and fed to an RoI pooling layer. Here, each proposed region and corresponding feature map is utilised to predict bounding boxes, classifications, and confidence scores. A visual representation of Faster R-CNN's architecture can be seen in Figure 2.8.

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

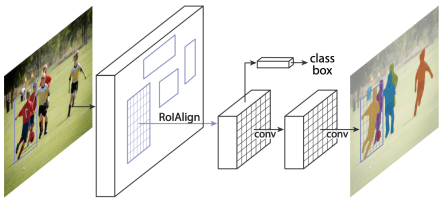


Figure 2.9 A visual representation of the changes made to Faster R-CNN to create Mask R-CNN. [39]

of additional convolutional layers which generate pixel predictions and instance mask outputs. See Figure 2.9 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 [3, 18, 60, 82], agriculture [20, 54, 75, 113], sports [13, 68, 73], astronomy [14], and nautical [44, 69] fields. As well as being well known, Mask R-CNN is also extremely reproducible. An official PyTorch implementation is available through Facebook AI Research's Github [104], whilst Matterport's Mask R-CNN implementation is most commonly utilised when working with Tensorflow (including in this project) [98].

### 2.3.6 Utilising Computer Vision in Cetacean Conservation

NEED TO ADD IN Lee et al, Backbone Alignment and Cascade Tiny Object Detecting Techniques for Dolphin Detection and Classification

POSSIBLY REWRITE THE IS WHOLE SECTION

The idea of utilising statistical methodology and machine learning in a marine cetacean space has, in recent years, been gaining popularity, with multiple papers being published in this area. Karnowski *et al.* propose using Robust PCA to subtract background from underwater images to help identify captive bottlenose dolphins, and track their movements through multiple distinct areas, allowing researchers to annotate pool positions 14 times faster than before [47]. Bouma *et al.* provide a system focusing on metric embedding learning to photo-id individual common dolphins, achieving top-5 accuracy scores of around 93% [10]. Further, Quiñonez *et al.* propose a CNN based system to detect four classes: `dolphin`, `dolphin_pod`, `open_sea`, and `seabirds` [76].

Outside of statistical theory and academic papers, multiple systems have been developed, or are currently in development, to aid cetacean researchers in the photo-id process. The first of these, known as FinScan, was developed in 2000 [43]. This is a semi-automated photo-

id assistant whereby the user imports images taken during fieldwork which FinScan then attempts to create a trace of the fin in the image. Users may manually edit this trace however if it is not exact (this feature was developed due to frustration with barnacles attaching to fins in the area where FinScan was developed). This trace of the fin is than checked against a local Microsoft Access database to determine close matches which are presented to the user. Before images are imported into FinScan they must be manually cropped, sharpened, and rotated by the user. Rotation of the image is especially important, and the FinScan algorithm is not rotation invariant. Further to this due to FinScan being an old piece of software, whilst it is freeware it is no longer readily available. Anyone who wishes to use it must procure a copy from someone else, there is no central repository for downloading. Issues with running the software on newer systems may also be present.

Similarly to FinScan, FinBase is a photo-id database management system developed by NOAA Fisheries [28]. 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 matches 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 [35] provides automated identification of new images based on those already in the attached database. Like FinScan, users of DARWIN trace around the leading and trailing edges of the fin they wish to identify. These edges are stored in a database as a set of evenly spaced points approximating the outline of the fin which is then used for identification.

Photo-ID Ninja[1] is a system currently in development which aims to photo-id individuals based on pigmentation commonly found on fins of the New Zealand common dolphin (*Delphinus* spp.)[31]. Here, pigmentation is used due to the low chance of other prominent markings becoming present on individuals. Matching is performed via the Euclidean distance between the input image and the catalogue of known individuals, which is then sorted by ID and validated using 5 fold cross-validation [9]. Current reported accuracies for this pigmentation matching are a top-1 accuracy of 90.6%, top-5 accuracy of 93.6%, and a mAP of 80.8%. This pigmentation matching is still in development however, although Photo-ID Ninja does currently provide a bulk cropping mechanism to aid in the speed up of manual identification. Users can provide a batch of images taken directly from the field to the system, which then outputs a zip file of cropped fins which can then be manually identified.

[1] Photo-ID Ninja: photoid.ninja

Work undertaken by Georgetown University and Google in the area of cetacean photo-id has also provided promising results [55, 64, 95]. The system, which utilises Google's Cloud Auto ML framework, can quickly identify bottlenose dolphins from Australia's Shark Bay . This system shows users the top-200 closest matches along with their confidence scores utilising both the leading and trailing edges of the fin. It is reported that this system saves Georgetown University's cetacean team around 4500 hours per year, highlighting the need for systems such as these to researchers in this field. However, this system does not link to a backend database to log matches found - this must be done manually by the researchers. Further, any new individuals which need to be added to the system, or indeed if the system was to be redeployed to a new area, then all training of the underlying model must be performed by Google engineers rather than locally by the researchers who wish to utilise the system. Further, fins to be identified must be inputted one by one, no batch input function exists.

In the past few years, systems utilising CNNs have started to enter the photo-id space. HappyWhale[2] 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[3] by user Jinmo Park to identify patterns present on the tailstock of the humpbacks [46], utilising elements of ArcFace [24] and DeepFace [90] to do so. Users interact with HappyWhale through their website, uploading images of the tailstocks encountered. The HappyWhale system then attempts to identify the individual before presenting back to the user. If the user provides location data, HappyWhale also keeps track of this to produce travel maps for the individuals, as humpback whales are known to travel vast distances in their lives. Success rate for HappyWhale varies greatly, from 99% for "good to high quality" images to 50% for full fins at 50x50px. HappyWhale struggles with partially obscured tailstocks however, and work is currently ongoing in this area [15].

Of all the systems in use or development today, one of the largest and most well known is FlukeBook[4], a fully automatic photo-id system for bottlenose dolphins, humpback whales, and sperm whales (*Physeter macrocephalus*). This system is part of a wider network of animal identification tools based on Wildbook, an open source software framework developed by non-profit organisation WildMe to facilitate the introduction of artificial intelligence into the conservation space [6]. Within FlukeBook there are two main photo-id algorithms; CurvRank and FinFindR.

CurvRank is an algorithm developed by Weideman *et al.* [102] which automatically identifies the trailing edge of the fin and represents this as a set of ordered coordinates.

[2]HappyWhale: happywhale.com
[3]Kaggle competition: kaggle.com
[4]FlukeBook: flukebook.org

Each coordinate point then has a circle of radius $r$ placed upon it, before being transformed horizontally. The curvature at this point for a given $r$ value is then defined as the ratio of the area under the curve againsy the area of a square around the curve of length $2r$. This allows for the definition of the trailing edge of the fin to be rotation invariant.

FinFindR is an algorithm developed by Thompson *et al.* which allows for inputted images containing bottlenose dolphins to be identified. FinFindR works with uncropped images of the whole area unlike other systems which require just the fin to be in the image. FinFindR then automatically detects the dolphin and crops it out, saving a new image. Cropping can be performed on either the whole body or on the dorsal fin only. This cropped image is then passed to a canny edge detector which creates an embedding of the trailing edge of the fin. This embedding is mapped into a high dimensional space based on work in FaceNet [85], with clustering of individuals achieved using Ward's variance minimising clustering [101]. Reported accuracies for FinFindR currently stand at a top-1 accuracy of 88%, top-5 accuracy of 94%, and top-50 accuracy of 97% [91]. It should be noted however that FinFindR has currently only been tested on bottlenose dolphins and work is still ongoing. The code for FinFindR can be found on GitHub [5].

## 2.4 Fine-Grained Visual Categorisation

Whilst the world of coarse-grained visual categorisation has mostly been solved, research is now focusing on more fine-grained visual categorisation. Using this project as an example, the detection of dolphins in an image and classifying them at a coarse-grain level as `dolphin` is relatively easy with current computer vision systems, with the vast majority of work being setup and hyperparameter tuning. However, being able to finely classify these dolphins with individual identifiers is a much more difficult task, as is all other fine-grained categorisation tasks.

EXPAND MASSIVELY

### 2.4.1 Fine-Grained Datasets

One major issue with these tasks is the lack of available datasets. Unlike coarse-grained datasets such as ImageNet [23], these fine-grained datasets must be labelled by domain experts. As such, only a few of these datasets currently exist. The Caltech-UCSD Birds-200-2011 dataset (an updated version of the original Caltech-Birds-200 dataset [103]) is a dataset of 200 different bird species [97]. Similarly, the Stanford Dogs dataset contains

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

images of 120 different species of dog [48]. Outside of animals, the Women's Fashion: Coats dataset details the diversity within women's clothing at a fine-grained level [25]. As can be seen, these datasets mainly focus on identification at a species level rather than an individual-within-a-species level like this project. This is the main motivation behind the creation of the Northumberland Dolphin Dataset as part of this project...

EXPAND ALSO

# Bibliography


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

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

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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

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

[12] 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.

[13] 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.

[14] 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.

[15] Cheeseman, T. (2019). Ted Cheeseman Happywhale presentation-WMMC-Rise of the Machines.pdf.

[16] 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.

[17] 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.

[18] 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).

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Di, W., Wah, C., Bhardwaj, A., Piramuthu, R., and Sundaresan, N. (2013). Style Finder: Fine-Grained Clothing Style Detection and Retrieval. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 8–13, OR, USA. IEEE.

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

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

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

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

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

[31] 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.

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

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

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

[35] 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.

[36] 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.

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

[38] 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.

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

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

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

[47] 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.

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

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

[50] 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.

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

[52] 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.

[53] 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.

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

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

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

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

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

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

[65] 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.

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

[67] 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.

[68] 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.

[69] 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.

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

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

[72] 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.

[73] 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.

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

[75] 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.

[76] 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.

[77] 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.

[78] 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.

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

[80] 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.

[81] 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.

[82] 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.

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

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

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

[86] 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/.

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

[88] 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.

[89] 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.

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

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

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

[93] 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.

[94] 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.

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

[96] 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.

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

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

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

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

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

[102] 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.

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

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

[105] 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.

[106] 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.

[107] 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.

[108] 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.

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

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

[111] 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.

[112] 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.

[113] 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.

[114] 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.