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 [44]. 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 [12, 62]. 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.

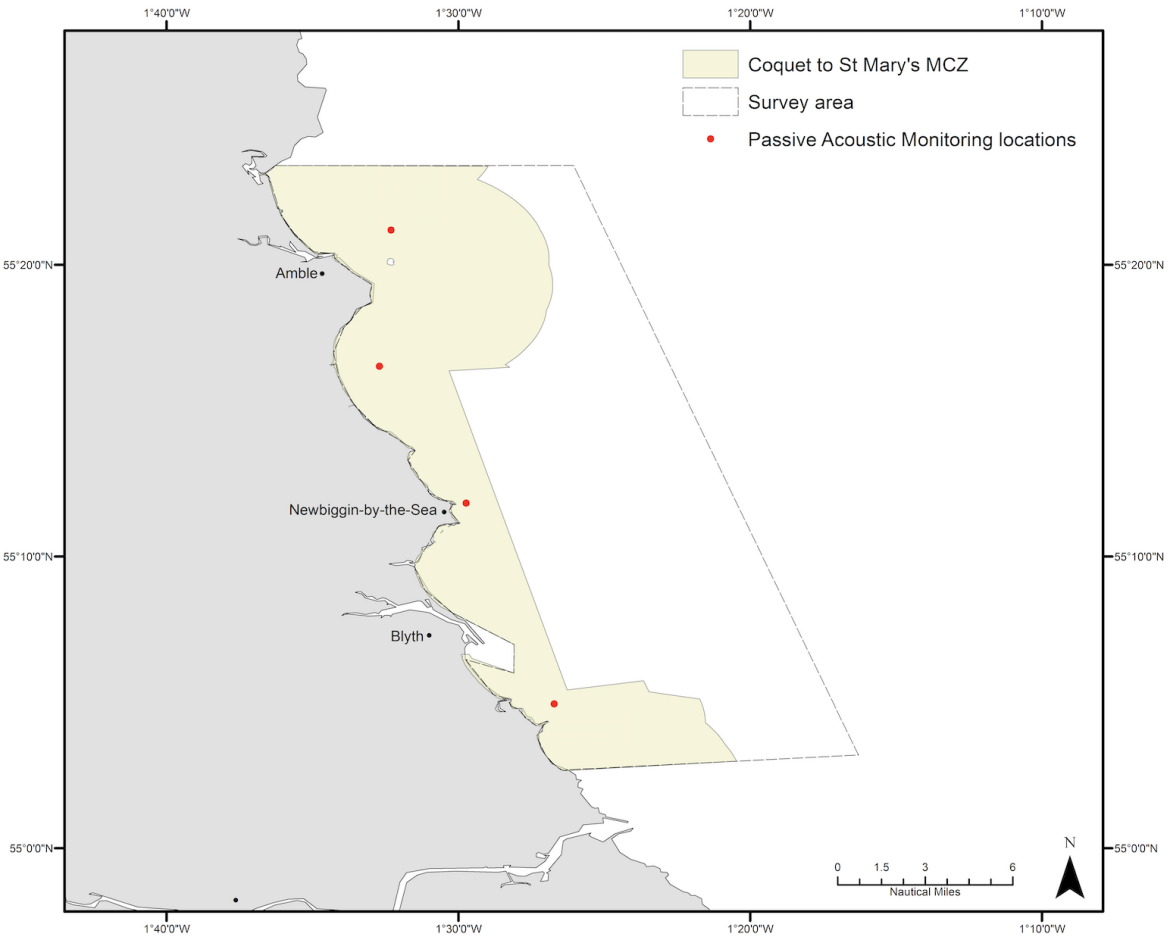

Figure 1.1 Map of the survey area, Northumberland, UK, from St. Mary's Lighthouse in the south to 25nm above Coquet Island in the north.

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

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

water images, and aerial drone footage). Methodologies incorporating these models are be 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 [16, Hammond and Berggren] 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 [59]. 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 [33]. As the field of deep learning increases in scope every day, this literature review 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 [27]. 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 [43] 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 [47] 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 datasets such as MNIST [34], Caltech-256 [20], and ImageNet [14] 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 [46].

EXPAND!!! 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 [54], batch normalisation [28], stochastic gradient descent with warm restarts [42], mixup [65], 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 [23], 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. $K = 2$) , the algorithm would most likely 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 [**?**]. 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 [6].

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 [42], Momentum [48], RMSProp [57], Adam [31], and AMSGrad [50]. 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 [11, 13].

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 [38], although it's effect was not fully realised until 1986, when Rumelhart *et al.* discussed the advantages to using backpropagation over other learning approaches [53]. Whilst multiple attempts have been made to improve the original algorithm [4, 35–37, 45], 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 is one area where deep learning has really excelled. Concepts such as CNNs have quickly become commonplace for solving computer vision tasks, in most cases replacing the need for specialised hand-crafted pipelines. The era of CNNs in computer vision started with LeNet [34], which was developed to recognise hand written digits on cheques. Before this, character recognition had been achieved with hand engineered feature spaces which machine learning models would learn to classify. With LeNet, these hand engineered features were made redundant, as the CNN could learn the optimal representation of these features from the characters themselves.

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 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. Whilst this basic principle may seem simplistic, CNNs now form the basis of most computer vision research, being utilised in facial recognition, autonomous vehicles, and fine-grain visual categorisation.

EXPAND THE LAYERS WITH MATHS

Convolutional Layers

The convolutional layer is the workhorse of the CNN, performing the vast majority of the operations required. This layer will operate over the whole input image provided using a kernel, sliding over the image spatially computing dot products. These kernels usually start looking for low level features such as line groups first, working up to more complex shapes the deeper the layer is. This allows for one image to become of stack of filtered images, or feature maps. These feature maps show how much of the feature matches at each individual pixel location.

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. Note pooling only affects the width and height of the input, not the depth (an RGB colour scheme has a depth of 3). This reduction inevitably leads to a reduction in the amount of information available in the input; this is advantageous however as it leads to less computational complexity for subsequent layers 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 [8], and stochastic pooling [64].

Fully Connected Layers

Fully connected layers, usually preceded by multiple convolutional and pooling layers in most architectures, are layers which each neuron has a connection to all other neurons in the previous layer. Thus, their activation function can be computed through a matrix multiplication with a bias weighting. These layers convert the list of feature maps into 1-d feature vectors, which can then either be considered a map in its own right for further

processing [32] or as a category for classification as the last layer of the network [19]. For example in a CNN which classifies either `x` or `y`, the final fully connected layer would have two neurons, one for each of the respective classifications.

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, mostly through trial and error, 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 the vast majority of computer vision tasks.

As discussed previously in Section 2.3, LeNet [34] 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 [32]. 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 [54], max pooling [8], and ReLU activation functions; the most popular non-linear activation function currently in deep learning, especially in computer vision [26].

The activation function is responsible for deciding which neuron in the layer passes it's value to the layer below by computing the weighted sum of the inputs and passing the result through a non-linear function. ReLU's non-linear function returns 0 for any negative value, or for any positive value x, it returns x. This can be written as $f(x) = max(0,x)$. 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 [56]. 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% [25]. Shallower versions of ResNet exist, such as ResNet50 and ResNet101, which are 50 and 101 layers deep respectively.

2.4 Object Detection Algorithms

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.

2.4.1 Region Proposal Networks

The first, known as a Region Proposal Network (RPN), attempts to find image regions likely to contain objects of given classes. Training data is usually provided in the form of bounding boxes drawn around objects of interest and labelled with the corresponding class. These RPN detections can be relatively fast, using a selection search [58] gives around 2000 region proposals in only a few seconds on a CPU.

Selection search is most commonly used with the **R-CNN** object detection algorithm [19]. 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** [18]. 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 runtimes. This also means however that the runtime is dominated by how fast ROIs can be generated.

To fix this issue, **Faster-RCNN** was developed [52]. 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.

2.4.2 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 [51] and SSD [40].

2.5 Cetacean Object Detection

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 [29]. 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% [7]. Further, Quiñonez *et al.* propose a CNN based system to detect four classes: `dolphin`, `dolphin_pod`, `open_sea`, and `seabirds` [49]. It is thus clear from recent literature that there is great interest in speeding up the photo-id of cetaceans for tracking and conservation efforts.

Outside of newer deep learning object detection, cetacean classification in the world of marine biology is already aided through software such as DARWIN [55] and Wildbook [5]. These systems however require a large amount of human preprocessing and input which new deep learning systems would not require.

2.6 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.

2.7 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.

In general, semantic segmenters can be though 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 a 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`. 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 [22] or Facebook's Mask-RCNN [24].

Second, a Fully Convolutional Network (FCN) can be utilised for semantic segmentation [41]. An FCN learns pixel to pixel mappings without the need for region proposals, whsilt 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 [3] and DeepLab [10].

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 [9].

2.8 Instance Segmentation

TODO

2.8.1 Fine-Grained Datasets

One major issue with these tasks is the lack of available datasets. Unlike coarse-grained datasets such as ImageNet [14], 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 [61]) is a dataset of 200 different bird species [60]. Similarly, the Stanford Dogs dataset contains images of 120 different species of dog [30]. Outside of animals, the Women's Fashion: Coats dataset details the diversity within women's clothing at a fine-grained level [15]. 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...

2.8.2 Part Segmentation

Whilst fine-grained visual categorisation is still an area of new research, one of the most common approaches to tackling these problems 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 [66]. How this is to be achieved is still being explored, with some research focusing on a form of hierarchical part matching [63], some on alignment of objects to define a super-class shape [17], some utilising deformable part descriptors [67], and others using part localisation [39].

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] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

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

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

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

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

[Hammond and Berggren] Hammond, P. S. and Berggren, P. Cetacean abundance and distribution in European Atlantic shelf waters to inform conservation and management | Elsevier Enhanced Reader.

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

[23] 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.

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

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

[26] 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.

[27] 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.

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

[29] 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.

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

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

[32] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

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

[34] 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.

[35] Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. pages 498–515. Springer.

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

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

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

[43] 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.

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

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

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

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

[55] Stewman, J., Debure, K., Hale, S., and Russell, A. (2006). Iterative 3-D Pose Correction and Content-Based Image Retrieval for Dorsal Fin Recognition. In Campilho, A. and Kamel, M. S., editors, *Image Analysis and Recognition*, Lecture Notes in Computer Science, pages 648–660. Springer Berlin Heidelberg.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

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

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

[62] 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.

[63] 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.

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

[65] 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.

[66] 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.

[67] 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.