UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Automatic detection of beaked whale echolocation clicks via convolutional neural networks

Tomás Lourenço Marques Vicente Gueifão

Mestrado em Bioestatística

Trabalho de projeto orientado por:
Prof. Doutor Tiago André Lamas Oliveira Marques
Doutor Emmanuel Dufourq

2024

Acknowledgments

I would like to express my sincere gratitude to several individuals whose support and assistance have been invaluable in the completion of this master's thesis.

First and foremost, I extend my thankfulness to Kalliopi Gkikopoulou for her continuous assistance with Matlab. Her expertise and guidance were instrumental in navigating through various challenges encountered during the earlier stages of this thesis.

I also extend my gratitude to Natacha de Soto, Mark Johnson, and Leigh Hickmott for generously providing an additional set of annotations from tagged Blainville's and Cuvier's beaked whales. Their contributions significantly enriched the scope of this research.

I owe Emmanuel Dufourq an immense amount of gratitude for his invaluable insights and guidance on machine learning concepts, and their applications in the realm of bioacoustics. His mentorship has played a crucial role in moulding my perspective on the topic and how I approach it.

I would want to express my sincere gratitude to Tiago Marques for his continuous support and giving me the chance to be involved with the ACCURATE project. It was through this involvement that the idea for this thesis came to be, and I am grateful for the inspiration and resources provided. Furthermore, I am indebted to Tiago for his inspiring classes, which ultimately led me to choose Biostatistics as my master's degree. For that, I am forever grateful.

I am incredibly grateful to Daniel Nunes for all the help he provided me with the thesis' code. His expertise and willingness to lend a helping hand were instrumental in overcoming various technical hurdles. Without his support, this thesis would not have been possible.

Special thanks are also due to Francisco Soares for his invaluable assistance in creating one of the images utilised in this thesis. I am truly thankful for his readiness to lend a hand whenever I sought assistance with the image. His creativity and technical proficiency greatly enhanced the visual presentation of this thesis.

My heartfelt thanks extend to Clarisse Hetier, Andreia Mindouro, Inês Paixão, Sandra Cardoso, and Ricardo Barros for their companionship and support throughout the duration of my Master's degree. Their encouragement and camaraderie have made this journey all the more enriching.

Last but certainly not least, I extend my deepest gratitude to my family, particularly my parents, for their unwavering support and encouragement. Without their love and guidance, this

achievement would not have been possible.

Abstract

In ecological studies, over recent decades, biological datasets have increased rapidly, both in size and complexity. This emphasises the necessity for ecologists to have practical tools to analyse this abundance of data and an understanding of modern techniques. Here, machine learning plays an important role since it automates and streamlines the process, enhancing efficiency in analysing and understanding the large amount of data available. Using data collected primarily from digital acoustic tags (DTAGs) attached to Blainville's beaked whales (*Mesoplodon densirostris*), supplemented by additional information from Cuvier's beaked whales (*Ziphius cavirostris*), the goal of this thesis was to develop a modern machine learning model capable of automatically identifying the distinct echolocation clicks emitted by these species using deep learning techniques via convolutional neural networks (CNNs). Two distinct experiments were conducted. Firstly, to evaluate the models' classification capabilities. Secondly, an existing echolocation click detector tool was compared to the CNN models developed in this study. Each experiment included multiple scenarios, consisting of different dataset configurations and objectives, that were assessed using accuracy, recall, precision, and F1-score. The CNN developed in this study, designed as a binary classifier for detecting the presence or absence of Cuvier's beaked whales, achieved an F1-score of 93.28% when applied to 15.8 hours of data, which corresponded to 27568 correctly classified audio segments and 802 incorrectly classified. While the CNN model developed for Blainville's beaked whales achieved an F1-score of 84.12% when applied to 3.4 hours of data, which corresponded to 5544 correctly classified audio segments and 521 incorrectly classified. The difference in performance between Cuvier's and Blainville's beaked whale models could be attributed to data scarcity. By developing these CNN models, the aim was to identify echolocation clicks and, furthermore, to provide a steppingstone for accurately estimating population densities, which can be done by resorting to methodologies such as cue counting.

Keywords: Bioacoustics, Convolutional Neural Networks, Machine Learning, Deep Learning, Beaked whale

Resumo

A última parte do século XX marcou uma mudança sem precedentes, na qual se transitou de tecnologias mecânicas e analógicas para tecnologias eletrónicas digitais, permitindo maior acessibilidade a equipamentos digitais a preços mais acessíveis. Esta revolução do mecânico e analógico para o digital transformou a capacidade de armazenamento dos equipamentos e a partilha de dados. Consequentemente, em campos de estudo como a ecologia, os conjuntos de dados biológicos têm crescido tanto em tamanho quanto em complexidade nas últimas décadas. Isto enfatiza a necessidade de os ecologistas terem acesso a novas ferramentas práticas capazes de analisar esta abundância de dados e de uma compreensão abrangente das técnicas modernas empregadas. No contexto da proteção da vida selvagem, torna-se essencial desenvolver uma compreensão profunda das populações atuais de várias espécies e das tendências nos seus números ao longo do tempo. Avaliar eficazmente a abundância de animais num determinado local e determinar se as suas populações estão a aumentar ou a diminuir representa um desafio considerável para os cientistas. Aqui, o machine learning desempenha um papel crucial, pois automatiza e agiliza o processo, aumentando a eficiência na análise e compreensão da grande quantidade de dados disponíveis. Os mamíferos marinhos, especialmente golfinhos e baleias, dependem fortemente do som, utilizando a emissão de sinais acústicos e a interpretação dos seus ecos, um processo designado por ecolocalização, para aspetos essenciais das suas vidas, como a alimentação, a comunicação e a navegação no ambiente. No entanto, devido à crescente poluição sonora marítima resultante de diversas atividades humanas, muitas dessas espécies têm sido seriamente afetadas nas suas atividades essenciais, levando, em alguns casos, a consequências fatais. Para avaliar a extensão destes impactos e criar medidas de conservação eficazes, a aplicação de técnicas de monitorização acústica passiva torna-se imperativa. A monitorização acústica passiva envolve a utilização de dispositivos de gravação subaquáticos para captar os sons produzidos pelos animais, permitindo aos cientistas monitorizar as suas atividades sem os perturbar. Consequentemente, existe uma procura constante por metodologias capazes de monitorizar as tendências populacionais destes animais. Uma metodologia que se tem destacado é a estimativa da densidade de monitorização acústica passiva (PAMDE). Um método dentro do PAMDE é a contagem de indícios (cue counting), através do qual os cientistas aproveitam os sons emitidos pelos animais para estimar com precisão a sua densidade. Tendo em conta a aplicação crescente de métodos como

o PAMDE, o objetivo desta tese centrou-se em desenvolver um modelo moderno de machine learning focado na identificação automática dos cliques de ecolocalização distintos emitidos por duas espécies de baleias-de-bico, nomeadamente baleias-de-bico de Blainville (*Mesoplodon densirostris*) e baleias-de-bico de Cuvier (*Ziphius cavirostris*), utilizando técnicas de deep learning através de redes neurais convolucionais (CNNs). Numa fase posterior, os cliques identificados poderão ser utilizados para aplicar métodos de PAMDE, permitindo assim uma estimativa mais precisa das populações dessas espécies. A recolha de dados para estas espécies foi feita através da fixação de tags acústicas digitais (DTAGs). Os dispositivos DTAG foram fixados em animais com ventosas. Este procedimento envolveu a utilização de uma vara especializada equipada com a DTAG na sua extremidade, que, quando uma baleia emergia para respirar, era aderida à região dorsal da baleia. A duração da gravação de dados pelo dispositivo dependia da sua capacidade de memória e da frequência com que recolhia áudio. Uma vez separada do animal, a DTAG flutuava à superfície da água e, para auxiliar na sua recuperação, emitia um sinal de rádio de frequência muito elevada (VHF), possibilitando a localização e recuperação das tags. O som recolhido pelas DTAGs foi analisado e transformado em espectrogramas, os quais correspondem a representações visuais das frequências de um sinal ao longo do tempo, mostrando a intensidade das diversas frequências como diferentes cores ou níveis de brilho. Esta transformação é crucial para a análise de sinais acústicos complexos, permitindo a visualização e identificação de padrões, como os cliques de ecolocalização emitidos pelas baleias. A recolha de dados para as baleias-de-bico de Blainville abrangeu o período de 2003 a 2017, enquanto para as baleias-de-bico de Cuvier os dados foram recolhidos de 2003 a 2013. As baleias-de-bico de Blainville foram etiquetadas em locais distintos, nomeadamente nas Bahamas e em El Hierro, nas Canárias. Da mesma forma, as baleias-de-bico de Cuvier foram etiquetadas em duas regiões geográficas distintas: Ligúria, em Itália, e sul da Califórnia. Os dados referentes a estas espécies na tese foram disponibilizados pelo projeto ACCURATE, inicialmente começando a trabalhar apenas com baleias-de-bico de Blainville e depois suplementando esses dados com informações adicionais das baleias-de-bico de Cuvier. A principal razão para a escolha destas espécies residiu no facto de serem as que continham a maior quantidade de dados recolhidos. Para treinar as CNNs, os espectrogramas dos cliques de ecolocalização foram utilizados como dados de entrada. Uma vez que estas redes neurais são especialmente eficazes para o processamento de dados visuais, o objetivo era que, durante o treino, as CNNs aprendessem a reconhecer padrões específicos nos espectrogramas que correspondiam aos cliques de ecolocalização das baleias-de-bico de Blainville e de Cuvier. O intuito era que as redes neurais demonstrassem a capacidade de identificar as vocalizações produzidas por estas baleias de uma forma automática e precisa. Este processo de treino envolveu a exposição das redes a milhares de exemplos de espectrogramas manualmente classificados, permitindo que a rede ajustasse os seus parametros para minimizar a diferença entre as suas previsões e os dados reais. No decorrer da tese, foram realizadas duas experiências distintas para avaliar a eficácia dos modelos desenvolvidos. Primeiramente, foi conduzida uma avaliação

das capacidades de classificação dos modelos de CNN. Esta fase foi crucial para determinar se as CNNs podiam identificar corretamente os cliques de ecolocalização em condições distintas. Em segundo lugar, foi feita uma comparação entre uma ferramenta existente de deteção de cliques de ecolocalização e os modelos de CNN desenvolvidos neste estudo. Este passo foi essencial para validar a eficácia dos novos modelos em relação às soluções previamente disponíveis. A comparação envolveu o uso dos mesmos conjuntos de dados para garantir uma avaliação justa e objetiva das capacidades dos modelos de CNN em identificar os cliques de ecolocalização. Cada uma das experiências incluiu múltiplos cenários, consistindo em diferentes configurações de conjuntos de dados e objetivos específicos. Estas configurações variaram em termos de complexidade e características dos dados, permitindo uma análise abrangente da performance dos modelos em diversas situações. Para avaliar os resultados, foram utilizados vários critérios de desempenho, incluindo a exatidão (accuracy), a sensibilidade (recall), a precisão e uma métrica combinada conhecida como F1-score. O F1-score, em particular, é uma métrica importante que considera a precisão e o recall, fornecendo uma visão equilibrada da eficácia dos modelos. A CNN desenvolvida neste estudo, projetada como um classificador binário para detetar a presença ou ausência de baleias-de-bico de Cuvier, alcançou um F1-score de 93,28% quando aplicada a 15,8 horas de dados, o que correspondeu a 27.568 segmentos de áudio corretamente classificados e 802 incorretamente classificados. Enquanto que o modelo de CNN desenvolvido para as baleias-de-bico de Blainville alcançou um F1-score de 84,12% quando aplicado a 3,4 horas de dados, o que correspondeu a 5.544 segmentos de áudio corretamente classificados e 521 incorretamente classificados. A diferença no desempenho entre os modelos das baleias-de-bico de Cuvier e das baleias-de-bico de Blainville pode ser atribuída à escassez de dados. Através do desenvolvimento destes modelos de CNN, o objetivo foi não só identificar cliques de ecolocalização de maneira automática, mas também fornecer uma base sólida para futuras aplicações de métodos de PAMDE. A identificação precisa dos cliques é um passo crucial para a estimativa de densidades populacionais, permitindo que os cientistas apliquem metodologias como a contagem de indícios em conjuntos de dados vastos mais rapidamente e com um grau de eficácia igual ou superior.

Palavras-Chave: Bioacústica, Redes Neurais Convolucionais, Machine Learning, Deep Learning, Baleia de bico

Contents

List of Figures

List of Tables

List of abbreviations

1D – 1 dimension
2D – 2 dimensions
3D – 3 dimensions
Adam – Adaptive moment estimation
AI – Artificial intelligence
BANTER – Bio-acoustic event classifier
CITES – Convention on international trade in endangered species of wild fauna and flora
CNN – Convolutional neural network
DFT – Discrete Fourier transform
DTAG – Digital acoustic tag
EWA – Exponentially weighted average
ICI – Inter-click interval
IUCN – International union for conservation of nature
MNIST – Modified national institute of standards and technology
PAM – Passive acoustic monitoring
PAMDE – Passive acoustic monitoring density estimation
RELU – Rectified linear unit
RGB – Red, green, and blue
RMSprop – Root mean square propagation
STFT – Short-time Fourier transform
VHF – Very high frequency

Chapter 1

Introduction

The closing years of the twentieth century saw a transition from the mechanical and analog electronic technologies to digital electronics. This transition was subsequently termed as the 'Digital Era', an epoch in which technological progress enabled an unparalleled rise in the pace and extent of knowledge transfer within the economy and society [54]. This technical advancement permitted greater accessibility and affordability of digital equipment, as well as data storage and sharing. Consequently, this has resulted in an unprecedented influx of data, that vastly outnumbers the capacity of labelling the datasets derived from it. Still today, most labeling is done by hand ([34, 69, 108]), making this method increasingly unsustainable over time.

Machine learning has taken off in recent years for a variety of reasons [96]. As previously noted, the rapid rise in processing power and data storage has allowed for more data to be processed and analysed at a faster rate, allowing machine learning to be applied to more complex scenarios and provide more accurate predictions. Another key element is the increased data availability, a consequence of the 'Digital Era'. This data serves as valuable input for training machine learning models, enabling the anticipation of outcomes and the automation of tasks.

In the field of ecological studies, biological datasets have experienced significant growth both in sheer volume (size) and intricacy (complexity) over recent decades. A more intricate dataset may encompass multi-dimensional data, integrating spatial, temporal, and genetic information [74]. As these datasets expand, there is increasingly a need not only for practical approaches capable of making sense of this data abundance, but also for a comprehensive understanding of the employed techniques. Applications of machine learning, particularly in computer vision (a branch of artificial intelligence that allows systems and computers to interpret digital images, videos, and other visual inputs to provide meaningful information), provide useful insights and methods that may be used to ecological research with success [157]. By incorporating and adapting research from these diverse domains, ecological studies can benefit from advanced analytical techniques and innovative approaches to address complex biological datasets. Climate change and its current and imminent effects on the Earth constitute a pressing concern within

the global community. Some of the primary problems emerging from this situation include deforestation, significant loss of biodiversity, an increase in the occurrence of extreme weather, and sea level rise due to de-icing [3]. To properly understand the root causes of these problems, how to foresee them, and how to guard against them, it is hypothesized that extensive data collection is necessary, in order to construct models potentially capable of making useful verdicts [18]. Machine learning can be used to automate the systematic collection and analysis of large volumes of data [91]. Valuable applications include the use of machine learning models capable of predicting severe flooding through extreme quantile regression [109], automatically detecting calls of one of the world's rarest mammal species, such as the Hainan gibbon [29], and automatically identifying bird species using audio recordings of their vocalisations [102] are just a few examples of how they can aid in ecological studies.

In order to safeguard wild animals, it is necessary to possess a comprehensive understanding of current populations and the trends in their numbers. Effectively assessing the abundance of animals in a given location and determining whether their populations are on the rise or decline currently poses a significant challenge for scientists. Consequently, there is a perpetual quest for improved methodologies to gather this vital information. Among the innovative approaches, passive acoustic monitoring density estimation (PAMDE) has emerged as a promising technique [140]. The application of PAMDE is particularly valuable when studying species that are very elusive. An illustrative method within PAMDE is cue counting, wherein scientists leverage the sounds emitted by animals to accurately estimate their density, marking a noteworthy advancement in the science of wildlife conservation [88].

Marine mammals, particularly dolphins and whales, heavily rely on sound for vital aspects of their lives, including feeding, communication, and navigating their environment [115]. These intelligent creatures use echolocation, a process wherein they emit acoustic signals and derive a sense of their surroundings from the resulting echoes. Through echolocation, they adeptly locate prey, engage in social interactions, and gain a comprehensive understanding of the complexities of their surroundings [7]. Unfortunately, the escalating levels of maritime noise pollution pose significant threats to their acoustic communication [33]. The surge in human-induced underwater noise disrupts the marine environment, leading to detrimental consequences such as disorientation, collisions with ships, and compromised communication, hindering crucial activities like mating calls [32]. To gauge the extent of these impacts and work towards effective conservation measures, the application of passive acoustic monitoring techniques becomes imperative. By employing these methods, scientists can estimate population densities, offering valuable insights into the existing numbers of marine mammals and helping comprehend the ramifications of human-induced challenges on their populations.

This thesis will concentrate on a specific type of mammals known as whales, the Blainville's beaked whales. Blainville's beaked whale, scientifically classified as *Mesoplodon densirostris*, is a toothed whale belonging to the family *Ziphiidae* (family of beaked whales) within the order

Cetacea (whales, dolphins, and porpoises), characterized by its distinctive beak-like snout [115]. These marine mammals play a vital part in the marine ecology not only by aiding in the maintenance of the food web through their eating habits as apex predators, but also by influencing nutrient cycling in the ocean [45]. Whales can therefore be used as indicators of the health of marine habitats and as sentinels for pollution and other environmental consequences [38]. Cetaceans are particularly long-lived, highly specialised creatures that achieve sexual maturity at an older age and have restricted fertility, with just one calf at a time. These features render them unable of rapid adaptation and thus vulnerable to anthropogenic impacts [121].

Beaked whales (*Ziphiidae*), are a highly speciose family, well suited for deep water environments, yet despite being such a species-rich family, they are one of the least known large mammals [22]. They usually aggregate in smaller groups, averaging between 3 or 7 animals, and display cryptic surface habits, including very short surfacing intervals and the ability to dive for incredibly long periods of time, making them quite challenging to spot while making field transects [9]. Blainville's beaked whales are renowned for their characteristic deep dives, engaging in foraging activities at depths ranging from 222 to 1885 metres, with intermittent shallower dives observed [144]. As adults, Blainville's beaked whales can reach lengths of 4.5 to 6 metres and weights of 820 to 1040 kilograms [115]. Despite the fact that there are no substantial variations in body lengths between genders, sexual dimorphism is most obvious. Mature males are identified from females and juveniles by the presence of a pair of enormous tusk-like teeth that protrude from their gum tissue and are visible from the outside mouth (Figure 1.1).

Figure 1.1: Photograph of a male Blainville's beaked whale. Figure adapted from https://iwdg.ie/blainvilles-beaked-whale/

Blainville's beaked whale conservation status has recently undergone a revision regarding its conservation status. According to the International Union for Conservation of Nature (IUCN) Red List of Threatened Species (IUCN 2020), Blainville's beaked whales have been reassessed

from its previous classification of "Data Deficient" to "Least Concern". Despite the fact that there is only limited information available concerning its global population size and none on the trends in abundance for this species, the decision to reclassify was made because Blainville's beaked whale has the most widespread distribution of any species in the *Mesoplodon* genus [115]. Blainville's beaked whales possess a widely distributed range in tropical and warm temperate waters across all major ocean basins (Figure 1.2), there being areas where it is frequently encountered and is locally common [81, 2]. These whales are often found in deep water habitats, although they favour areas with topographical complexity such as undersea canyons, shelf edges, and seamounts [83, 156]. This may be because food is concentrated around in these areas [12, 65]. A pronounced variation in water depth also plays a significant role in the habitat of Blainville's beaked whale, since it may allow the ability to feed in both the deep scattering layer and near the sea bottom on a small spatial scale [82, 6]. Nonetheless, like many other species of cetaceans, Blainville's beaked whale is listed under CITES Appendix II [2], indicating that it is at risk of extinction if commercial activities are not strictly regulated. This highlights the importance of continued conservation efforts to ensure the survival of this species for future generations.

Densities of *Mesoplodon* beaked whales appear to be fairly low, despite the fact that substantial fluctuation in densities suggests the existence of "hotspots" in some locations, Barlow ([10]) discovered a relatively low density of Blainville's beaked whales near Hawaii, having estimated 1.17 whales/1000km^2. In contrast, when using data from fixed passive acoustic sensors – devices stationary in the environment designed to detect and record sounds – the estimation of Blainville's beaked whale density on the Atlantic Undersea Test and Evaluation Center range involved accounting for cue detection probability (in this context referring to the sound produced by the animal), and the proportion of false positive detections ([90]). This conversion yielded an average density of 22.5 to 25.3 whales/1000km^2 over a 6-day period. Through the use of bioacoustics it is possible to study the animal sounds as a tool for estimating population densities providing non-invasive means of detecting the presence of the whales. Fixed hydrophone arrays, stationary underwater microphone systems, are employed to capture and analyse whale sounds in marine environments. These arrays detect and record vocalizations and echolocation clicks, offering valuable data on whale presence, distribution, and behavior. By studying these recorded sounds, species can be identified, the movements tracked, and the population densities estimated. In cases when visual surveys or physical traps are difficult, costly, or dangerous, passive acoustics provides an alternate survey option. Whales, regardless of size, possess streamlined bodies and muted coloration, which blend perfectly with the ocean environment. This, coupled with their frequent submersion at significant depths and low density over large areas of ocean, makes them visually cryptic creatures. They do, however, make audible and identifiable noises that can be used to measure abundance and density [90]. Even if visual or trap-based approaches are viable options, passive acoustic solutions remain advantageous in such scenarios.

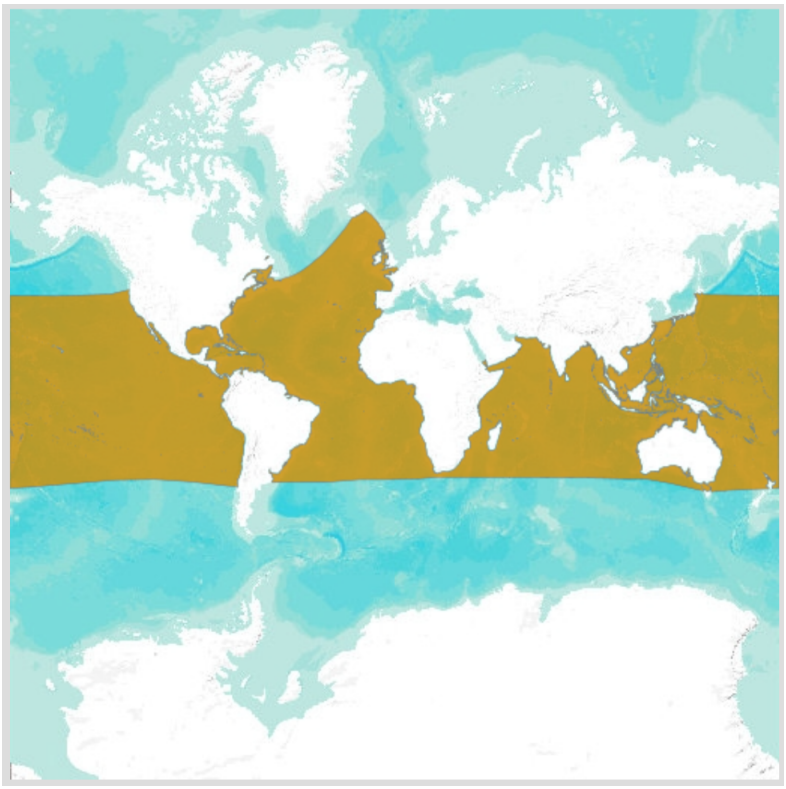


Figure 1.2: Blainville's Beaked Whales geographic range, highlighted in yellow. Figure adapted from IUCN (International Union for Conservation of Nature) 2012. Mesoplodon densirostris. The IUCN Red List of Threatened Species. Version 2023-1

Animals that make loud or frequent sounds may be detectable at greater distances than by other methods. Furthermore, underwater passive acoustic surveys are less affected by weather conditions, and are more favourable to automating data collection, since they allow for larger amounts of data to be analysed, when compared to conventional observational methods. This data collection approach proves particularly effective in marine studies, as sound encounters less resistance underwater than in air, leading to enhanced transmission effectiveness. Moreover, because light diminishes with depth, sound is a better method not just for communication between these individuals but also for hunting. Throughout foraging dives, Blainville's beaked whales utilise echolocation, emitting sounds for navigation and interpreting the returning echoes, to locate prey. Over different stages of their foraging, these whales emit two unique types of sounds in the form of clicks. During foraging dives, Blainville's beaked whales emit search clicks, specifically designed to locate prey by interpreting the echoes produced when the signal hits the target, the same way a sonar functions. These clicks have an *inter-click intervals* (ICIs) ranging from 0.2 to 0.4 seconds, which refer to the time between consecutive echolocation clicks, and take the shape of an upsweep in sound, like a musical note rising in pitch, designated as a frequency

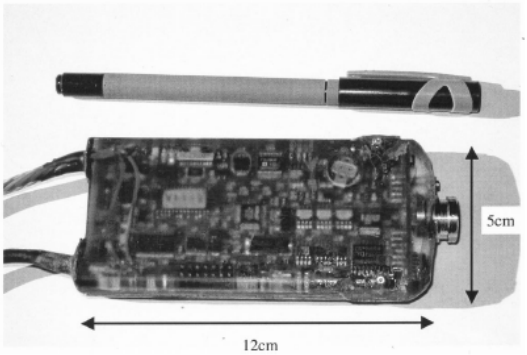


Figure 1.3: Photograph of the inside of a DTAG and its respective dimensions. Image adapted from [59]

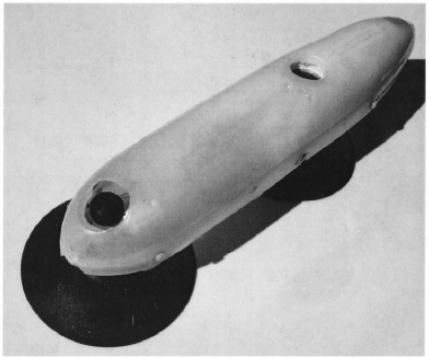


Figure 1.4: Photograph of a complete DTAG with two suction cups to stay fixed on the whale. Image adapted from [59]

modulation upsweep (modulation rate of roughly 110 kHz/ms). This sound has a pulse, short burst or sound signal duration, of 270 μs and a unique frequency range from 26 to 51 kHz. The intensity of the sound within this range is very concentrated, indicated by a -10 dB bandwidth (this negative value highlights the specific frequency range where the sound is most intense). During the last stage of prey capture, Blainville's beaked whales generate buzz clicks. These clicks are quick bursts, lasting only 105 microseconds, and they don't have the same rising-and-falling pitch pattern as the search clicks. Instead, they cover a broader frequency range, from 25 to 80 kHz or even higher, with a concentrated intensity in this range indicated by a -10 dB bandwidth [58]. These two click are suited for different purposes, with search clicks tailored for target detection and classification, while buzz clicks are specialized for efficient prey capture in a densely populated auditory environment.

In order to track Blainville's beaked whales vocal behaviour while applying non invasive passive acoustic monitoring techniques, a new archival tag, denominated DTAG [59], developed to monitor marine mammals and their responsiveness to sound during the dive cycle was administered (Figures 1.3, 1.4).

The aim of this thesis is to investigate, develop, and deploy a machine learning model capable of processing all the audio data recorded by the DTAG latched onto the Blainville's beaked whales, automating the process of identifying the presence of one of the two forms of clicks noises that it emits, the search clicks. By doing so, it is possible to estimate the density of animals as demonstrated by Marques *et al.*, in [89].

Chapter 2

Introduction to Machine Learning and Deep Learning

2.1 Introduction

This chapter introduces machine learning, the history about the origin of the term, and how it differentiates itself from adjacent concepts. Standard terminology, as well as common procedures utilised in the implementation of machine learning algorithms will also be discussed. Following the introduction to machine learning, the sub-field of deep learning will be addressed in greater depth in order to offer all of the details required to comprehend the methods used in this thesis.

2.2 Introduction to Machine Learning

When machine learning is discussed, it is important to firstly understand that it differs from conventional computer programming. In conventional programming, a programmer establishes specific rules, algorithms, that the computer program adheres to in order to generate outputs. In contrast, in machine learning, a computer program is designed to function through data-driven processes. Instead of developing precise algorithms to perform a task, like it is done in conventional programming, machine learning models use iterative processes to adjust their parameters based on the data inputted.
In 1959, Arthur Samuel played a pivotal role in popularizing the term by developing a computer program capable of playing the 2-player board game, checkers [129]. Arthur Samuel defined machine learning as the study field enabling computers to learn without explicit programming. Illustrating this with Samuel's checkers example, conventional programming entails providing data on board piece placement and detailing each possible move's rules. In this approach, the computer follows a set of textual rules to play the game accurately. In the context of machine learning, the process involves providing data on the position of each game piece on the board.

However, instead of coding specific rules for each piece's movement, a set of observations from past games – instances of move sequences – is provided. The process of being able to play the game autonomously using earlier examples, rather than relying on written code, is what has led to the characterization of the machine as "learning".
A later and more popular machine learning definition was Tom Mitchell's which deemed that a computer is considered to be learning when it gets better at certain tasks through experience [97]. This experience means learning from examples or data from the past. The tasks are the goals of the program, and success is measured by how well the computer achieves these goals. According to Mitchell, a computer "learns" when it becomes more skilled at its tasks with increasing experience, building on the core concept laid out by Arthur Samuel. These definitions are now regarded to be complimentary. As a result, machine learning can be defined as a field that focuses on the learning aspect of artificial intelligence [21], designing algorithms that "learn" relationships between data and improve performance on given tasks through pattern recognition, mimicking the human brain's learning process but using data rather than explicit programming.
From historical milestones to recent breakthroughs, machine learning has demonstrated its revolutionary capabilities [125, 73, 17]. In the 1950s, Arthur Samuel pioneered machine learning by developing an algorithm capable of playing checkers [129]. This marked an early milestone where machines demonstrated the ability to learn from past examples. Subsequent decades saw incremental progress, with notable advancements in the 1990s when Yann LeCun showcased a machine learning algorithm, more precisely a convolutional neural network, that was able to recognise handwritten digits [77]. Despite this achievement, the concept of neural networks only really took off in 2012 when it was used to win the ImageNet challenge [73]. This challenge, featuring a vast database of millions of images, demonstrated neural networks' ability to accurately identify a wide range of images. In 2020, a groundbreaking machine learning achievement emerged with the release of GPT-3, a state-of-the-art model designed for language comprehension and usage [17]. What sets GPT-3 apart is its remarkable ability to learn through conversational interactions, resembling communication with a person, without the need for explicit instructions or examples. This unique teaching approach enabled GPT-3 to proficiently perform diverse tasks and grasp subtle nuances in human language. Figure 2.1 illustrates a timeline of the mentioned machine learning breakthroughs, until the present time.

2.2.1 Artificial Intelligence vs Machine learning vs Artificial Neural Networks vs Deep Learning

This section seeks to highlight the distinctions among key concepts such as artificial intelligence, machine learning, deep learning, providing overall explanations for each term. Further exploration of some of these concepts will be undertaken in subsequent sections.
Artificial intelligence (AI) is a vast field which encompasses a variety of techniques, such as

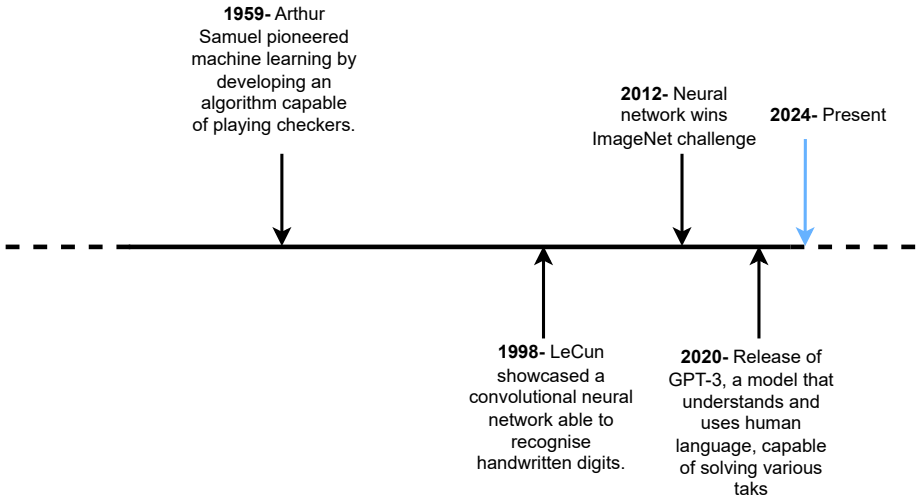


Figure 2.1: Timeline illustrating some breakthroughs in machine learning throughout the years.

machine learning [97], deep learning [46], computer vision [117], among others, with the goal of enabling computers to perform tasks that typically require human intelligence. In machine learning, we find methods like supervised and unsupervised learning [46], additionally, deep learning also utilises techniques such as convolutional neural networks [106]. Computer vision focuses on topics like image recognition [70] and object detection [49]. The main goal of AI is thus to develop computer algorithms that possess capabilities, such as recognizing patterns, learning from examples/data, problem-solving, decision-making, and adaptability to new situations [21, 127], which are typically attributed to a human level of intelligence, involving intricate thought processes associated with cognitive functions. Machine learning is a branch of artificial intelligence, which focuses on developing algorithms capable of improving their performance by identifying patterns in new data. Through exposure to past examples, these algorithms learn and adapt without explicit coding instructions, enhancing their abilities and enabling strong performances on previously unseen data. Artificial neural networks (sub-section 2.3.1), are a machine learning technique used for tackling distinct challenges within the field of AI, which comprise a set of algorithms designed to process and analyse data. However, this functioning is done in a way that closely resembles the intricate functioning observed in biological neural networks [37]. Deep learning can be summarised as a subset of machine learning that comprises artificial neural networks with multiple components. Deep learning is commonly employed to address challenges associated with more complex datasets. This topic will be discussed in a more detailed manner in section 2.3. Figure 2.2 illustrates how these concepts are connected to one another.

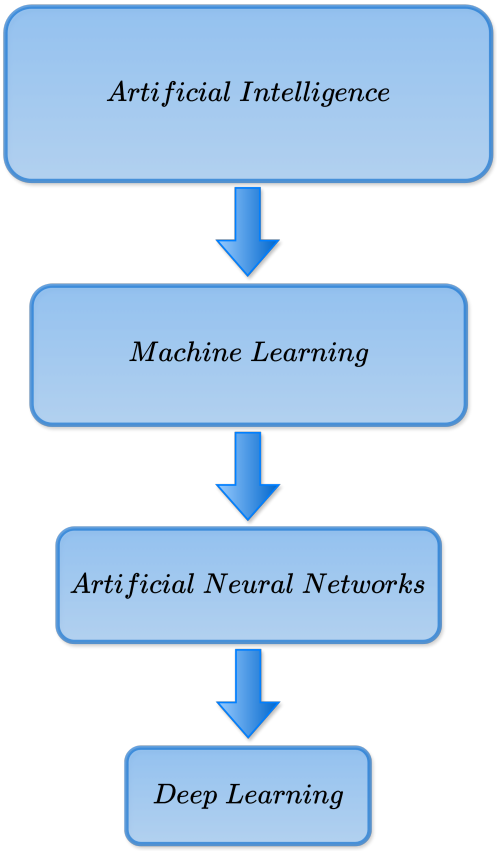


Figure 2.2: Diagram representing the domain of artificial intelligence and the fields that fall within its realm.

2.2.2 Steps to implement a Machine Learning model

Dataset terminology

To construct a machine learning model, it is necessary to first comprehend the terminology used. For any machine learning model, a dataset is required to start processing and learning from the data. A *dataset* is a structured file containing a collection of raw data, all of which is related to a particular subject. In a (tabular)dataset an entire row is called an *example*, *instance*, or *data point*. An *example*, corresponds to a single observation or set of data about the subject being studied. Each of these examples are in turn composed of one or more *features*, sometimes including single or multiple targets as well. A *feature*, also know as *inputs* or *attributes*, corresponds to a single column of data. These represent the variables or attributes that provide information about the example. The *target*, the feature of the dataset intended to be clearly understood and explained, is the variable intended to be predicted for each example based on the features applied to that same example. In most cases, a dataset comprises multiple exam-

ples and their respective targets. For instance, in a dataset focused on student performance, an example corresponds to a single row, representing the unique information about one student. Each piece of information about the student, such as the hours spent studying and attendance, is a feature. The target, in this scenario, is the final grade, serving as the variable the machine learning model seeks to predict based on the provided features. Therefore, a machine learning model can be described as one that is constructed to map input features to their designated targets [98].

Classification and Regression

Features can be characterised as *quantitative* or *categorical*. Quantitative features take on numerical values, such as a person's age, income, or the value of a house. In contrast, categorical features are qualitative and represent different categories. They take on one of K different classes, such as a person's gender ("male" or "female"), manner of transportation ("car", "bus", "train"), or education level ("high school", "bachelor's degree", or "master's degree").

Typically, problems with a quantitative target are referred as *regression* problems/tasks, while those with a categorical target are often termed *classification* problems/tasks. In classification problems, the target is also known as a *class* [56]. When a classification problem involves two classes, it is termed binary classification. If the problem has more than two classes, it is labeled as multiclass classification.

The objective of a classification model is to accurately predict the classes for as many examples as possible, while a regression model's goal is to predict values as closely as possible to the targets in each example. Both classification and regression problems are subdivisions of supervised machine learning problems.

Supervised and Unsupervised learning algorithms

There are two generally utilized machine learning algorithms, each appropriate for addressing different types of problems, categorised as supervised, and unsupervised, depending on the manner the examples are provided during the learning process. This thesis exclusively concentrates on techniques that utilize supervised learning algorithms. Thus, a more general definition of unsupervised learning will be provided, accompanied by additional bibliographic references for those with a deeper interest in exploring the theme.

Supervised learning algorithms begin with a dataset $D = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $i = 1, \ldots, n$, where each $\mathbf{x}_i$ represents a d-dimensional feature vector (example) and y_i denotes the corresponding target. Essentially, the supervised algorithm operates on a dataset in which each example is characterised by known feature values and their corresponding targets. In supervised learning, the target is commonly referred to as the *label*.

Assuming that these examples are drawn from an environment P, unknown to the neural network of interest,

$$(\mathbf{x}_i, y_i) \sim P \tag{2.1}$$

the goal of a supervised learning algorithm is finding a mapping function, denoted as f, capable of accurately predicting the label based on the provided features. In essence, the aim is for $f(\mathbf{x})$ to closely approximate y for every feature-label pair $(\mathbf{x}, y)$ sampled from P (as shown in Equation 2.2).

$$\hat{y} = f(\mathbf{x}), \quad \text{where} \quad \hat{y} \approx y \tag{2.2}$$

In the case of *unsupervised* machine learning algorithms for every example, $i = 1, \ldots, n$, the d-dimensional feature vector is known, however there is no associated target, y_i. Instead unsupervised learning algorithms aim to extract patterns in the dataset and categorise individual examples to said categories. Unlike supervised learning, there is no predetermined correct output for the algorithm to learn from. Instead, it autonomously discovers information and structures within the data. Some of the most common unsupervised learning tasks are clustering, association, and anomaly detection [56, 50].

Data splitting

When working with models in machine learning, particularly in the context of supervised learning, datasets are commonly split into different subsets: the training, validation, and testing sets. The training set, a subset of the main dataset, serves as the input for training the model, allowing it to learn patterns within the data. This set consists of labeled examples, where both input features and their corresponding target values are provided to the model during the learning process – this is the *training phase*. The validation set, comprising a distinct subset of the dataset, serves to evaluate the model's performance during the training phase. This set utilises independent examples, unseen by the model in the training set, to fine-tune various aspects of the model, such as hyperparameters and the architecture. *Hyperparameters*, external settings for the model not learnt from the data, are essential for controlling the machine learning model's algorithm's behaviour [46]. The *architecture* of the model represents the model's structure. It is in the validation set that the optimal values for hyperparameters and the most effective model architecture are identified. The testing set is an independent subset of the dataset reserved for the final evaluation of the model after all the training phase, including training and validation, is complete. The data in the testing set is used to provide an unbiased estimate of the model's capacity of generalisation with new, unseen data – this is the *testing phase* [87]. Test data (unseen data), should not be used for learning *parameters* (features of the model that obtain values through the algorithm training using the training set) or hyperparameters of the model [15].

Following training and fine-tuning of hyperparameters, the model is evaluated on the test set to determine the error of the resulting model when applied to new, unseen data – the *generalisation error*, or *test error*. It is crucial not to use the test set data during training and validation to ensure a reliable estimate of the generalisation error, as otherwise, it might become overly optimistic and unreliable [127]. Thus, the model is optimised during the training phase with the training and validation data, and evaluated during the testing phase, with the test data [158]. Although the percentage of data allocated for training, validation, and testing can vary between studies, based on dataset size and model characteristics, a common split is around 60-80% for the training set and 10-20% for both the validation and testing sets.

2.3 Deep Learning

2.3.1 Artificial Neural Networks

As previously defined, the emulation of tasks typically requiring a human level of intelligence is pursued within the field of artificial intelligence. In line with this, artificial neural networks, also known as neural networks, are based on mathematical models inspired by how human cognition or neural biology works. The structure of a biological neuron, which serves as an information messenger transmitting signals both intracellularly through electrical transmission and intercellularly via chemical transmission [99], closely resembles that of a perceptron, the fundamental unit of logic within an artificial neural network. For its operation, a biological neuron is configured in three essential components: dendrites, cell body/soma, and axon (fig. 2.3).
A biological neuron operates by gathering inputs from neighboring neurons or sensory organs through its dendrites, which are branched extensions designed to receive signals from adjacent neurons [99]. The cell body, also known as the soma, sums the incoming signals. Upon reaching a sufficient input, the cell transmits a signal, treated as binary, along its axon to other cells. This binary nature results from the fact that, at a given instant, the cell either attains the energy threshold, leading to signal firing, or falls short, resulting in the signal not propagating further [37].

2.3.2 Perceptron

The perceptron, also known as a neuron (artificial), unit, cell, or node, was introduced by Frank Rosenblatt in 1957 [125]. A perceptron is a binary algorithm that acts as a classifier (assigns examples to different classes based on specific features), by determining if something is true or false, 1 or 0. The goal of a perceptron is to map some example, $\mathbf{x}$, to a single target, or label, y (Figure.2.4).
In the context of a perceptron or a neural network, the model processes individual examples from

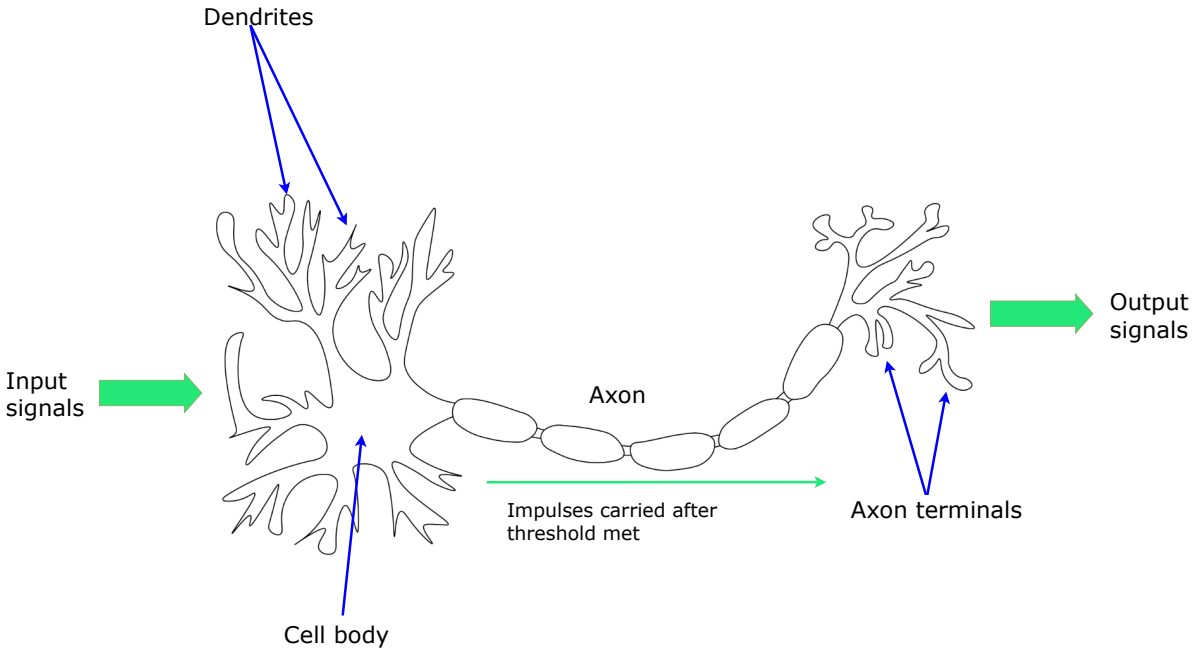


Figure 2.3: Illustration of a biological neuron with its main constituents portraying the flow of information, under the shape of electrical signals within the neuron and chemical between neurons.

the dataset, each represented as a feature vector, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, where x_i for $i \in \{1, 2, \ldots, n\}$ corresponds to each feature of the example. To assess the strength of the contribution of each input feature x_i on the perceptron's output y, a parameter designated by *weights* is used. Since certain features hold greater significance than others, given a specific task, the incorporation of *weights* in perceptrons, and neural networks in general, is crucial [37]. These weights, denoted by $\mathbf{w} = [w_1, w_2, \ldots, w_n]$, where $w_i \in \mathbb{R}$ for $i \in \{1, 2, \ldots, n\}$, serve as adjustable parameters, randomly initialised. This random initialisation is done with the purpose of optimisation, allowing the network to learn and adapt the weights during the training phase, with the goal of identifying which features are most important for the given task. In order to generate the perceptron's output value y, the example is aggregated through the dot product of each weight and their corresponding input feature, as demonstrated in equation (2.3).

$$\mathbf{w} \cdot \mathbf{x} = \sum_{i}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \tag{2.3}$$

The aggregate of equation (2.3) is then passed through a specific activation function, $f(x)$, the step function, resulting in binary outputs. Further details on the concept of activation function will be elaborated in subsection 2.3.3. Equation 2.4 presents the $f(x)$ function.

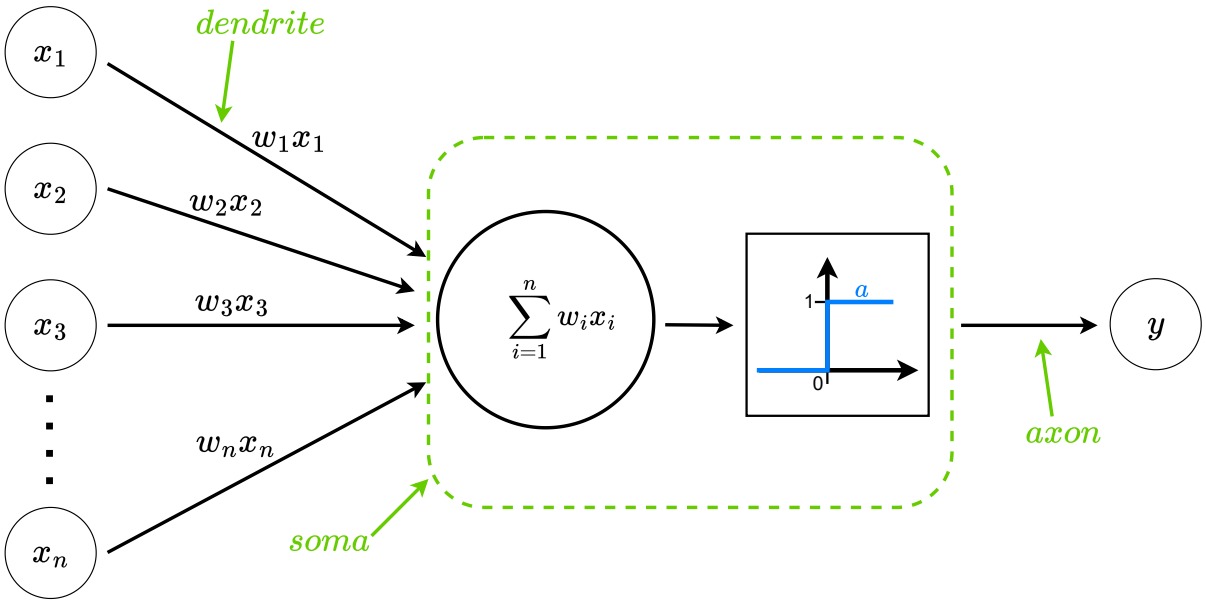


Figure 2.4: Illustration of a perceptron (a type of artificial neuron), indicating the similarities between a biological neuron and a perceptron structure through the green labels. The components of this perceptron include the example [x_1, x_2,..., x_n], the weights [w_1, w_2,..., w_n], the dot product of the weights and their corresponding feature inputs, a step activation function a, and an output y.

$$f(x) = \begin{cases} 1, & \text{if} \quad \sum_i^n w_i x_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

In figure 2.4, the process is illustrated, in which each i^{th} feature input x_i establishes a communication link with the corresponding i^{th} weight w_i, depicted by the black arrows. These links have the capability to either excite or inhibit the signal, depending on the patterns present within the dataset. The aggregation of all feature inputs and their corresponding weights is portrayed within the circle located in the green-highlighted region. Then, the step activation function $f(x)$, represented by the square, processes this aggregate to produce the model's output y. Thus, the perceptron represents a single layer network. Similar to a canonical linear function (see Equation 2.5), the introduction of a bias term b in the perceptron expands its capacity to encompass a broader range of linear classifiers – a method used to categorise examples into distinct classes using a straight-line approach. The addition of a bias allows the perceptron to determine where the function intersects the x-axis, giving it the ability to shift along the axis.

$$f(x) = ax + b \tag{2.5}$$

Equation 2.6 showcases the altered function $f(x)$, where the bias term was added.

$$f(x) = \begin{cases} 1, & \text{if} \quad \sum_i^n w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.6}$$

2.3.3 Activation Functions

In Equation (2.6), the step activation function was introduced, characterized by its binary output. This particular activation function gives the perceptron the capacity to effectively model linear classifiers. Although activation functions were briefly mentioned in the previous subsection, with the introduction of the step function, this section will go into further detail, providing a proper definition of the concept and exploring some of the different, existing types. An activation function, refers to the function applied to the input features and weights in a neural network. It serves as the threshold, deciding whether a neuron is 'activated' or 'not activated.' In other words, it emulates the firing process that occurs in the cell body of a biological neuron (see figure 2.3). In neural networks, several types of activation functions are employed. *Sigmoid*, *rectified linear units* (ReLUs), and *softmax* are some of the most often utilised activation functions in artificial neural networks.
Moving further, it is important to understand a key distinction between the terms "perceptron" and "neuron". Even though they are occasionally used interchangeably, this usage is not entirely accurate. Generally, "perceptron" specifically refers a type of artificial neurons that use a step activation function, while "neuron" serves as a broader term encompassing a wider variety of activation functions. Thus, from here on, "neuron" will be used, and "perceptron" will only be mentioned for instances where explicit reference to an artificial neuron with a step activation function is intended.
The *Sigmoid* activation function [152], presented in equation 2.7, is a nonlinear function and a variant of the logistic equation that can be used as an output unit in a binary classifier to determine the probability of $P(y = 1|x)$ [120].

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

The sigmoid function has the shape of an S curve (figure 2.5). Because its domain is the set of all real numbers within the range (0, 1) and its derivative has range (0, 0.25), the sigmoid function is also known as a squashing function. This means that if the input is a very big negative or positive number, the output will always be between 0 and 1. As a result, the sigmoid function can be employed to convert a real value into one that can be read as a probability.
The *Rectified Linear Unit* (ReLU) [100], presented in equation 2.8, is a prevalent activation function in deep learning models. Its appeal arises from its simplicity and efficiency in tackling the vanishing gradient problem, a topic that will be explained further in sub-section 2.6.1.

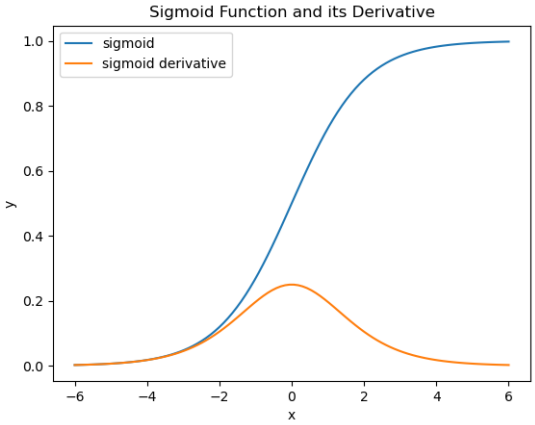


Figure 2.5: Plot of a sigmoid function and its corresponding derivative.

$$f(x) = max(0,x) \tag{2.8}$$

The ReLU function outputs the value of the input, if the input value is positive, and zero if the input value is negative. This can be visualised in Fig.2.6.

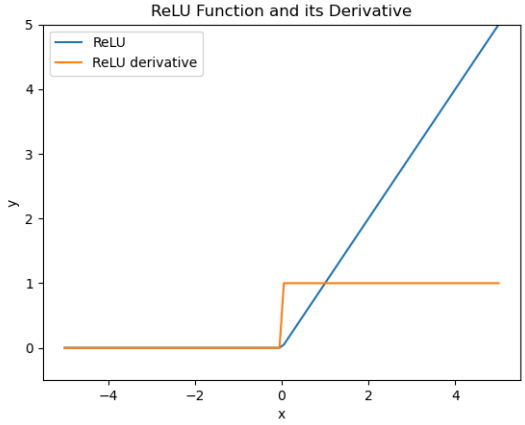


Figure 2.6: Plot of a rectified linear unit function and its corresponding derivative.

The *Softmax* activation function [15], as defined in equation 2.9, is a widely used function in neural networks. It transforms a vector of numbers into a vector of probabilities, ensuring that each probability falls between 0 and 1, and the sum of all probabilities equals 1. Softmax is particularly useful in multiclass classification problems, where it converts raw scores into probabilities representing the likelihood of an input feature belonging to each class.

$$f(x) = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}} \tag{2.9}$$

where, K corresponds to the number of classes, and $j \in \{1, \ldots, K\}$.

2.4 Multi-layer neural networks

In sub-section 2.3.2 the concept of the perceptron, a single layer network, was introduced. A perceptron was defined as a type of artificial neuron with the capacity of solving linearly separable problems. However, they are not good for handling non-linearly separable ones [25]. To address these more complex problems, networks must contain more than a single layer, a concept known as *multi-layer neural networks*. These networks can also be referred to as *feedforward neural networks* where the term "feedforward" is used because data is strictly directed forward [46]. Within these networks, feedback connections or loops are not present (a key feature present in a distinct type of artificial neural networks called Recurrent Neural Networks [148]), meaning that no neuron's output contributes to a neuron preceding it within the network structure. Consequently, the complexity of these models originates from the interconnection of its functions in a chain-like manner. For example, if a model consists of four functions connected in a chain structure $f(x) = f^{(4)}(f^{(3)}(f^{(2)}(f^{(1)}(x))))$, where the bracketed superscripts denote the layers, the function $f(x)$ or $f^{(4)}(\cdot)$, is more complex than the functions from the preceding layers. The greater the number of functions connected in a chain, the more layers the network has, and the greater the complexity. The length of these chains is referred to as their depth. Networks with numerous layers are termed "deep neural networks" due to their substantial depth.
Artificial neural networks are composed with three distinct types of layers. The first layer of an artificial neural network is called the *input layer*, where each of the example's features is provided into the neural network. The last layer of an artificial neural network is called the *output layer*, which outputs the label's predictions or the classifications made by the model. The layers that exist between the input and output layers are the *hidden layers*. The reason these layers are dubbed 'hidden' is because within a model, they perform complex computations, however their outputs are not directly observable during model usage.
Figure 2.7 depicts a two-layer neural network, containing one input layer, one hidden layer, and one output layer. When counting layers, the input layer is excluded. The following notations will describe the components present in feedforward neural networks:

- The input layer, represented by the transposed vector $\mathbf{x} = [x_1 \quad \cdots \quad x_i \quad \cdots \quad x_n]^T$, comprises values where each x_i for $i \in \{1, 2, \ldots, n\}$ corresponds to a feature of the example. Here, n denotes the total number of features present in the example.

- The activations of the hidden layer, represented by $\mathbf{a}^{(1)} = [a_1^{(1)} \quad \cdots \quad a_j^{(1)} \quad \cdots \quad a_m^{(1)}]^T$, where

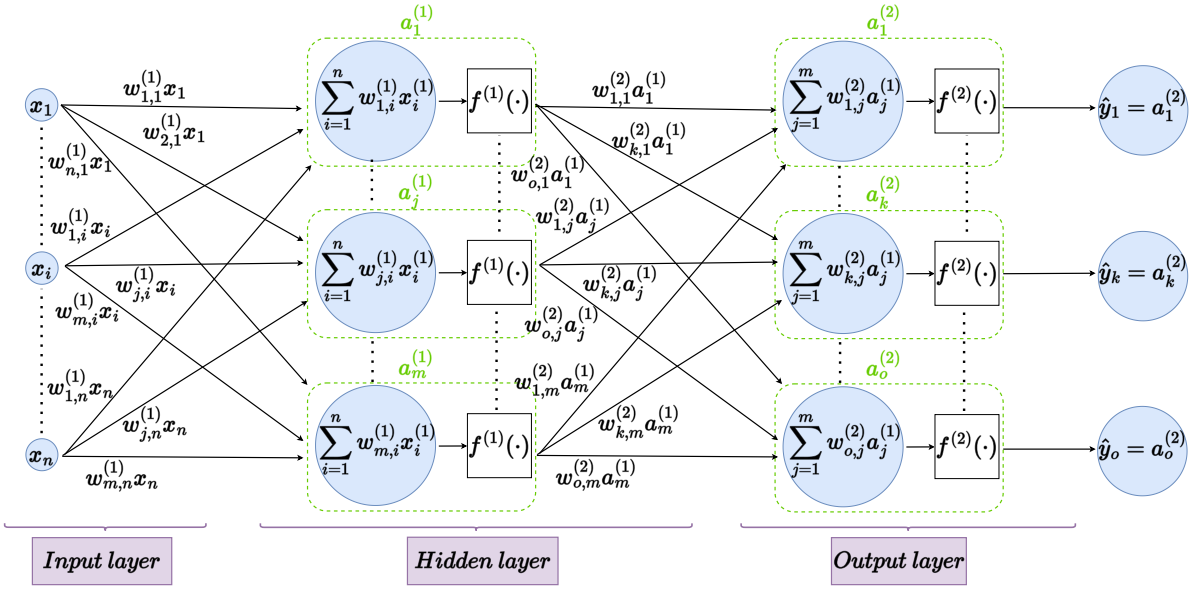


Figure 2.7: Graphical representation of a feedforward neural network with one hidden layer, where $\hat{y}$ represents the model's predictions.

the bracketed superscripts denote the layer, and each $a_j^{(1)}$ for $j \in \{1,2,...,m\}$ represents the activation of the j^{th} neuron in the hidden layer, are computed as the product of the sum of the weighted inputs from the input neurons, influenced by the established communication links, and the subsequent application of the activation function. Here m denotes the dimension, or the total number of neurons in the hidden layer.

- The activations of the output layer, denoted by $\mathbf{a}^{(2)} = [a_1^{(2)} \quad \cdots \quad a_k^{(2)} \quad \cdots \quad a_o^{(2)}]^T$, where each $a_k^{(2)}$ for $k \in \{1,2,...,o\}$ represents the activation of the k^{th} neuron in the output layer, are computed by taking the product of the sum of the weighted inputs from the input neurons, influenced by the established communication links, and subsequently applying the activation function. These activations yield the predictions made by the model, denoted by $\hat{\mathbf{y}} = [\hat{y}_1 \quad \cdots \quad \hat{y}_k \quad \cdots \quad \hat{y}_o]^T$, where $\hat{\mathbf{y}} = \mathbf{a}^{(2)}$. Here o denotes the dimension, or the total number of neurons in the output layer.

- The weights corresponding to the communication links between the input neuron $i \in \{1,\ldots,n\}$ and hidden neuron $j \in \{1,\ldots,m\}$, are represented as $w_{j,i}^{(1)}$. Here,the first subscript j denotes the index of the neuron in the current layer, which is the hidden layer, while the second subscript i denotes the index of the neuron in the previous layer, which is the input layer. These weights are contained in a weight-matrix denoted as $\mathbf{W}^{(1)}$.

- The weights corresponding to the communication links between the hidden neuron $j \in \{1,\ldots,m\}$ and output neuron $k \in \{1,\ldots,o\}$, are represented as $w_{k,j}^{(2)}$. Here,the first subscript

k denotes the index of the neuron in the current layer, which is the output layer, while the second subscript j denotes the index of the neuron in the previous layer, which is the hidden layer. These weights are contained in a weight-matrix denoted as $\mathbf{W}^{(2)}$.

- The activation functions of the hidden and output layers are represented by $f^{(1)}(\cdot)$, and $f^{(2)}(\cdot)$ respectively.

In the scenario where the hidden layer incorporates a bias $a_0^{(1)}$, assuming $a_0^{(1)} = 1$, the hidden layer activations would be represented as $\mathbf{a}^{(1)} = [a_0^{(1)} \quad a_1^{(1)} \quad \cdots \quad a_j^{(1)} \quad \cdots \quad a_m^{(1)}]^T$.

Given the information provided, a feedforward neural network can be expressed mathematically as a function. The total input signal of hidden neuron j, commonly denoted as $z_j^{(1)}$, where $j \in \{1, \ldots, m\}$, is determined by the product of features from the input layer and their respective weight parameters, as demonstrated in equation 2.10.

$$z_j^{(1)} = \sum_{i=1}^{n} w_{j,i}^{(1)} x_i \tag{2.10}$$

or in case the presence of a bias $a_0^{(1)}$, assuming $a_0^{(1)} = 1$, is considered,

$$z_j^{(1)} = \sum_{i=1}^{n} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)} = \sum_{i=0}^{n} w_{j,i}^{(1)} x_i \tag{2.11}$$

where, $w_{j,0}^{(1)}$ represents the weight which links between the bias and the hidden neuron.

After obtaining the total input signal of hidden neuron j, it is utilized to compute the activation of the j^{th} hidden neuron, denoted as $a_j^{(1)}$, by passing it through the corresponding activation function $f^{(1)}(\cdot)$, as shown in equation 2.12.

$$a_j^{(1)} = f^{(1)}(z_j^{(1)}) \tag{2.12}$$

These activations demonstrate the computation for a single neuron. However, it's possible to represent all activation functions as a vector $\mathbf{f}^{(1)}(\cdot)$ since the same activation functions are consistently applied within each layer. Although all the activation functions within a layer are the same, different layers can work with different activations. Hence, the activations of the hidden layer, denoted $\mathbf{a}^{(1)}$, can be expressed as the result of the product between the weights connecting the input layer and the hidden layer, represented by $\mathbf{W}^{(1)}$, and the input features $\mathbf{x}$, as described in equation 2.13.

$$\begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_j^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix} = f^{(1)} \left(\begin{bmatrix} w_{1,1}^{(1)} & \cdots & w_{1,i}^{(1)} & \cdots & w_{1,n}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j,1}^{(1)} & \cdots & w_{j,i}^{(1)} & \cdots & w_{j,n}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{m,1}^{(1)} & \cdots & w_{m,i}^{(1)} & \cdots & w_{m,n}^{(1)} \end{bmatrix} \begin{bmatrix} x_1^{(1)} \\ \vdots \\ x_i^{(1)} \\ \vdots \\ x_n^{(1)} \end{bmatrix} \right), \tag{2.13}$$

which is the same as writing

$$\mathbf{a}^{(1)} = \mathbf{f}^{(1)}(\mathbf{W}^{(1)}\mathbf{x}) \tag{2.14}$$

Similarly, the total input signal of output neuron k, denoted by $z_k^{(2)}$ for $k \in \{1, \ldots, o\}$ is calculated in the following manner:

$$z_k^{(2)} = \sum_{j=1}^{m} w_{k,j}^{(2)} a_j^{(1)} \tag{2.15}$$

or in case the presence of a bias $a_0^{(2)}$, assuming $a_0^{(2)} = 1$, is considered,

$$z_k^{(2)} = \sum_{j=1}^{m} w_{k,j}^{(2)} a_j^{(1)} + w_{k,0}^{(2)} = \sum_{j=0}^{m} w_{j,i}^{(2)} a_j^{(1)} \tag{2.16}$$

where, $w_{k,0}^{(2)}$ represents the weight which links between the bias and the output neuron.

The activation of the k^{th} output neuron is also calculated in a similar manner to equation 2.12,

$$a_k^{(2)} = f^{(2)}(z_k^{(2)}) \tag{2.17}$$

The entire feedforward neural network illustrated in figure 2.7, can thus be represented mathematically, when taking into account equations (2.10 - 2.17), resulting in

$$a_k^{(2)} = f^{(2)} \left(\sum_{j=1}^{m} w_{k,j}^{(2)} f^{(1)} \left(\sum_{i=1}^{n} w_{j,i}^{(1)} x_i \right) \right) \tag{2.18}$$

In a similar way to equation 2.13, the activations of the output layer, denoted $\mathbf{a}^{(2)}$, can be expressed as the result of the product between the weights connecting the hidden layer and the output layer, represented by $\mathbf{W}^{(2)}$, and the features from the hidden neurons $\mathbf{a}^{(1)}$, as described in equation 2.19.

$$\begin{bmatrix} a_1^{(2)} \\ \vdots \\ a_k^{(2)} \\ \vdots \\ a_o^{(2)} \end{bmatrix} = f^{(1)} \left(\begin{bmatrix} w_{1,1}^{(2)} & \cdots & w_{1,j}^{(2)} & \cdots & w_{1,m}^{(2)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{k,1}^{(2)} & \cdots & w_{k,j}^{(2)} & \cdots & w_{k,m}^{(2)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{o,1}^{(2)} & \cdots & w_{o,j}^{(2)} & \cdots & w_{o,m}^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_j^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix} \right), \tag{2.19}$$

which is the same as writing

$$\mathbf{a}^{(2)} = \mathbf{f}^{(2)}(\mathbf{W}^{(2)}\mathbf{a}^{(1)}) \tag{2.20}$$

2.5 Loss and Cost functions

In the previous section 2.4, the concept of feedforward neural networks was introduced, covering the operations involved in a multi-layer neural network from receiving the input example, defined as the vector $\mathbf{x}$, to generating the output predictions, defined as the vector $\hat{\mathbf{y}}$. However, when the model predicts an output, $\hat{y}$, this prediction may not correspond to the exact target, y. The difference between the actual target and the model's prediction is defined as the *error* [51]. To quantify this error with respect to a single training example, defined *loss functions* $\mathscr{L}(y,\hat{y})$ are used – mathematical functions designed to calculate the difference between the predicted and true outputs. There are several different loss functions that can be utilised in machine learning [124]. The decision on which of these functions is better to apply depends greatly on the nature of the problem and the type of model that is intended to be trained [4, 155].

While minimizing the error for each individual training example is important when training a model, the main objective is to reduce the average loss across all training examples. This average loss across all training examples is defined as the *cost function*, or *objective function* [46]. The cost function evaluates the performance of the parameters on the training set. Consequently, the main goal of a machine learning model is to minimise the cost function, in order to effectively discover the optimal set of parameters capable of yielding predictions closely aligned with the target values.

Categorical cross-entropy [110, 48] is a commonly used function in multi-class classification tasks. Its loss function, denoted as $\mathscr{L}_{CCE}$ and illustrated in equation 2.21, corresponds to a probabilistic log-likelihood.

$$\mathscr{L}_{CCE}(y,\hat{y}) = -\sum_i y_i log(\hat{y}_i) \tag{2.21}$$

where, y represents the true class label, $\hat{y}$ the predicted class label, y_i the probability of the i^{th} class according to the true class label, and $\hat{y}_i$ the the probability of the i^{th} class according to the predicted label.

The cost function of the categorical cross-entropy, denoted as J_{CCE}, is a broader concept that encapsulates the overall "cost" or "penalty" associated with the model's parameters. This function, illustrated in equation 2.22, calculates the average (or total) loss across all examples within the dataset.

$$J_{CCE}(y,\hat{y}) = \frac{1}{N}\sum_{i}^{N} y_i log(\hat{y}_i) \tag{2.22}$$

where, N represents the total number of examples in the dataset, y represents the true class label, $\hat{y}$ the predicted class label, y_i the probability of the i^{th} class according to the true class label, and $\hat{y}_i$ the the probability of the i^{th} class according to the predicted label.

2.6 Optimisation

Section 2.5, introduced cost functions. These functions measure the proximity or disparity between the model's predictions and the true labels, based on the examples within the dataset. However, merely evaluating the model's performance is not enough. What is needed is a mechanism capable of adjusting the parameters – namely, the weights and biases – to minimise the cost function and thus improve the model's performance. For this purpose, optimisation algorithms, known as *optimisers*, are applied. These, are responsible for updating the parameters of the neural network [103].

2.6.1 Backpropagation

Artificial neural networks underwent a significant transformation with the introduction of *backpropagation* [127]. This method revolutionized how neural networks evaluate the influence of each neuron and its corresponding weight pair on the network's error. Prior to backpropagation, artificial neural networks struggled with complex real-world and theoretical problems. This was primarily due to the lack of an efficient method for updating or adjusting the weights in large networks, a limitation that was highlighted in the 1969 report by Minsky and Papert [95].
The *backpropagation* method effectively addresses this limitation by making possible the evaluation of the derivatives of the network performance with respect to the weights. This method is based on the concept of relative error contribution, where each neuron and its weight pair influence the overall network error. Essentially, the error signal can be traced backward from the output layer, where it originates, to all the neurons that contribute to the error. This comprehensive understanding of how each neuron-weight pair affects the overall error is made possible through the implementation of *the chain rule* [78].
While the description of the backpropagation method is crucial, a mathematical demonstration of how this method is applied is also necessary, in order to fully understand it.

Taking into account the notations used in section 2.4, consider a dataset containing Q training examples, where the individual error, or *loss*, $\mathscr{L}^q$ for the q^{th} input example, with $q \in \{1,\ldots,Q\}$, can be computed. The network error function J, or *cost* function, can be expressed as,

$$J = \sum_{q=1}^{Q} \mathscr{L}^q \tag{2.23}$$

where,

$$\mathscr{L}^q = \mathscr{L}^q(\hat{y}_1,\ldots,\hat{y}_o) \tag{2.24}$$

in which $\mathscr{L}^q$ is represented as a differentiable function of the network outputs. In other words, the error associated with a particular input example, $\mathscr{L}^q$, varies in response to changes in the neural network's outputs $(\hat{y}_1,\ldots,\hat{y}_o)$. Backpropagation introduces a systematic way to compute the derivatives of the network's error function J, relative to the weights. With the assistance of equation 2.23, these derivatives can be depicted as the sum of the individual error derivatives, considering each input example, across the training set.

Consider, the analysis of the weight $w_{k,j}^{(2)}$ depicted in the feedforward neural network example, present in figure 2.7, for some $k \in \{1,\ldots,o\}$ and some $j \in \{0,\ldots,m\}$, to assess the derivative of $\mathscr{L}^q$. The total input signal $z_k^{(2)}$ of the output neuron k, presented in equation 2.15, represents the only intermediate factor influencing the individual error $\mathscr{L}^q$ in relation to $w_{k,j}^{(2)}$. As a result, the chain rule of differentiation is applied:

$$\frac{\partial \mathscr{L}^q}{\partial w_{k,j}^{(2)}} = \frac{\partial \mathscr{L}^q}{\partial z_k^{(2)}} \cdot \frac{\partial z_k^{(2)}}{\partial w_{k,j}^{(2)}}. \tag{2.25}$$

The notations present in this equation can be simplified to

$$\delta_k^{(2)} = \frac{\partial \mathscr{L}^q}{\partial z_k^{(2)}} \tag{2.26}$$

which corresponds to the error of neuron k with respect to $z_k^{(2)}$. This represents how much the neuron k needs to change in order to reduce the overall error of the network. And,

$$\delta_j^{(1)} = \frac{\partial \mathscr{L}^q}{\partial z_j^{(1)}}. \tag{2.27}$$

corresponding to the error of neuron j with respect to $z_j^{(1)}$, which represents how much the neuron j needs to change in order to reduce the overall error of the network.

Taking into consideration equation 2.15, it can also be defined

$$\frac{\partial z_k^{(2)}}{\partial w_{k,j}^{(2)}} = a_j^{(1)}. \tag{2.28}$$

which represents how much $z_k^{(2)}$ changes with respect to $w_{k,j}^{(2)}$.
Taking into account these new equations 2.26 and 2.28, the original equation 2.25 can be broken down into,

$$\frac{\partial \mathscr{L}^q}{\partial w_{k,j}^{(2)}} = \delta_k^{(2)} a_j^{(1)}, \tag{2.29}$$

representing how much the weight $w_{k,j}^{(2)}$ contributes to the individual error $\mathscr{L}^q$.
Given the definition of $a_k^{(2)}$ in (eq:2.17) and the dependency of $\mathscr{L}^q$ on $z_k^{(2)}$ solely through the output neuron activation $a_k^{(2)}$, the following expression can be formulated

$$\begin{aligned}
\delta_k^{(2)} &= \frac{\partial \mathscr{L}^q}{\partial a_k^{(2)}} \frac{\partial a_k^{(2)}}{\partial z_k^{(2)}} \\
&= \frac{\partial \mathscr{L}^q}{\partial a_k^{(2)}} \frac{\partial}{\partial z_k^{(2)}} \left(f^{(2)} \left(z_k^{(2)} \right) \right) \\
&= f'^{(2)} \left(z_k^{(2)} \right) \frac{\partial \mathscr{L}^q}{\partial a_k^{(2)}}.
\end{aligned} \tag{2.30}$$

In order to obtain this expression, the *chain rule* is utilised, as demonstrated in equation 2.25. The derivative of $\mathscr{L}^q$ with respect to the weight $w_{j,i}^{(1)}$ (linking the input layer to the hidden layer) can be found for any j in the range $\{1,\ldots,m\}$ and any i in the range $\{0,\ldots,n\}$ by applying the same procedure. Consequently, resulting in the following equation:

$$\frac{\partial \mathscr{L}^q}{\partial w_{j,i}^{(1)}} = \frac{\partial \mathscr{L}^q}{\partial z_j^{(1)}} = \delta_j^{(1)} x_i^{(1)}. \tag{2.31}$$

Furthermore, since $\mathscr{L}^q$ can describe a function of the total input to each of the o output neurons, then equation 2.27 can be rewritten as:

$$\delta_j^{(1)} = \frac{\partial \mathscr{L}^q \left(z_1^{(2)}, \ldots, z_o^{(2)} \right)}{\partial z_j^{(1)}}. \tag{2.32}$$

Considering the total input to each output neuron as defined in equation 2.32, the following expression for $\delta_j^{(1)}$ is derived by employing the chain rule twice and substituting equation 2.26 into the resulting expression:

$$\begin{aligned}\delta_j^{(1)} &= \frac{\partial \mathscr{L}^q}{\partial z_1^{(2)}}\frac{\partial z_1^{(2)}}{\partial z_j^{(1)}} + \cdots + \frac{\partial \mathscr{L}^q}{\partial z_o^{(2)}}\frac{\partial z_o^{(2)}}{\partial z_j^{(1)}} \\ &= \sum_{k=1}^{o}\frac{\partial \mathscr{L}^q}{\partial z_k^{(2)}}\frac{\partial z_k^{(2)}}{\partial z_j^{(1)}} \\ &= \sum_{k=1}^{o}\delta_k^{(2)}\frac{\partial z_k^{(2)}}{\partial a_j^{(2)}}\frac{\partial a_j^{(2)}}{\partial z_j^{(1)}}.\end{aligned} \tag{2.33}$$

Using equations 2.12 and 2.15, $\delta_j^{(1)}$ can ultimately be defined as:

$$\delta_j^{(1)} = f'^{(1)}\left(z_j^{(1)}\right)\sum_{k=1}^{o} w_{k,j}^{(2)}\delta_k^{(2)} \tag{2.34}$$

which is referred to as the *backpropagation formula*. Figure 2.8 illustrates this derivation which was done based on the example of a feedforward network with a single hidden layer, provided in section 2.4. However, it can be generalised to feedforward networks with any number of hidden layers.

The backpropagation algorithm can thus be summarised as follows:

1. Input the q^{th} training example into the feedforward neural network, where $q \in \{1,\ldots,Q\}$.
2. Forward propagate the input through the network, calculating the total input signal to each hidden neuron j using equation 2.10, and then applying the activation function to obtain the activations of the neurons in each layer (see equations 2.10 – 2.20).
3. Calculate the error in the output layer using the target output and the predicted output, by using equation 2.30 to evaluate the $\delta_k^{(2)}$ value for each neuron in the output layer.
4. Backpropagate the $\delta_k^{(2)}$-values and calculate $\delta_j^{(1)}$ using equation 2.34.
5. Find the derivatives of each individual error ($\mathscr{L}^q$) with respect to the weights $w_{j,i}^{(1)}$, using the chain rule (see equation 2.29 and equation 2.31).

The derivative of the total error with respect to the weights in a feedforward neural network, as per equation 2.23, can be computed across the entire training set by successively implementing the algorithm. This is expressed by

$$\frac{\partial J}{\partial \mathbf{w}} = \sum_{q=1}^{Q}\frac{\partial \mathscr{L}^q}{\partial \mathbf{w}}, \tag{2.35}$$

allowing backpropagation to intuitively map out the valleys and peaks of the entire *error surface*, a concept that will be further explained in the following sub-section.

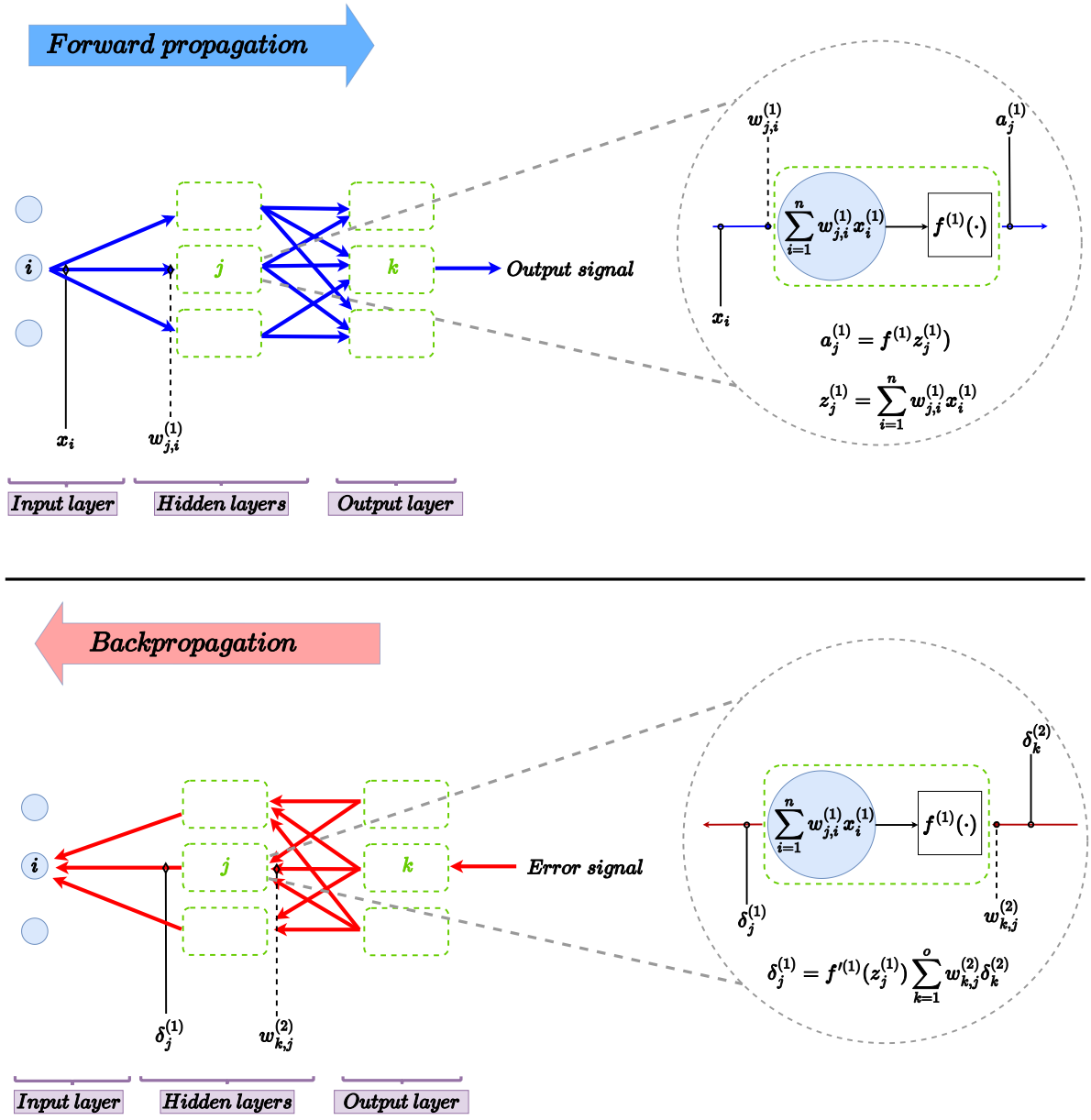


Figure 2.8: Figure representing a neural network where Backpropagation is applied. This process includes two different stages that are iterated until a local minimum is achieved, feedforward pass and a backwards pass.

2.6.2 Gradient Descent

In the previous sub-section 2.6.1, the concept of *error surface* was introduced. The term *error surface* can be conceptualised as a mapping of the cost function across the weight parameter

space, including bias if present. An example of an error surface is illustrated in figure 2.9.

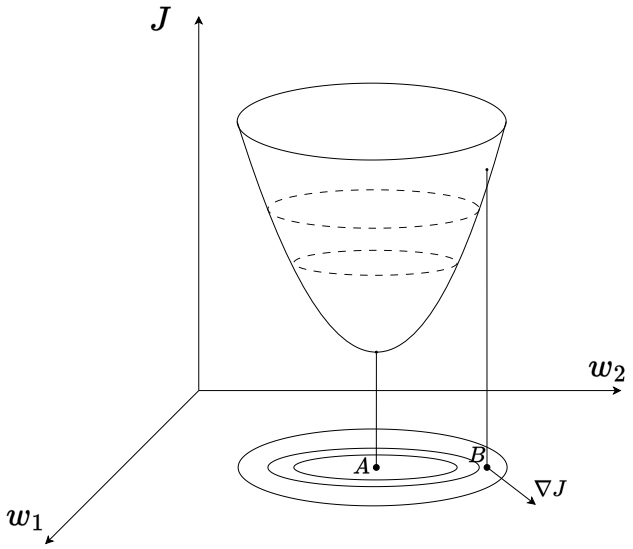


Figure 2.9: Figure representing an example of a cost function J generated by a neural network with the weight parameters w_1 and w_2. The global minimum of J is denoted by the point A. Point B on the error surface, denotes the local gradient represented by the vector ∇J.

In this figure, the error surface is depicted as a simple quadratic function, which provides a visual understanding of how the cost function behaves in response to different weight values. The lowest cost value is represented by the point A, which is also known as the *global minimum* of the surface. This point signifies the set of parameters, including weights, that minimise the cost, making it the most optimal choice for the given problem. The goal of training a neural network is to navigate this error surface efficiently, seeking out the global minimum. However, the dimensionality of the function may be more complex than a simple quadratic function. This complexity can result in an error surface with multiple peaks and valleys, as illustrated in Figure 2.10.

Each of the points present in one of the peaks or valleys is defined as a *critical point*. These points dubbed 'critical' because they represent points where the gradient equals zero, as per equation 2.37. If we consider the function $f(x_1, x_2, \ldots, x_n)$ as a function of n variables, the gradient vector ∇f is defined as follows:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right) \tag{2.36}$$

Thus, critical points are represented as,

$$\nabla J = 0 \tag{2.37}$$

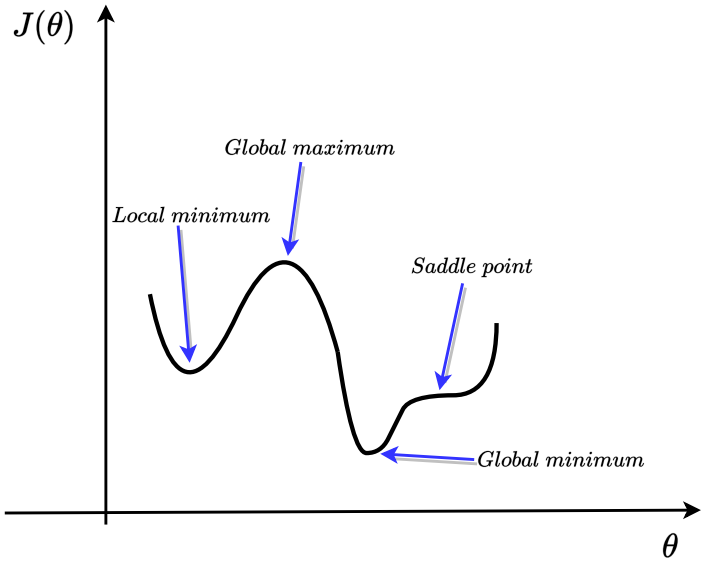


Figure 2.10: Nonlinear function with different sorts of critical points. A critical point is defined as the point at which the function's gradient is zero. There are several kinds of critical points. Local minima and maxima are defined as sites where the gradient of the function changes from negative to positive, and vice versa. A saddle point is defined as a position at which the function's gradient is the same on both sides of the stationary point. Global minima and maxima are the lowest and highest minimum and maximum points, respectively.

Critical points may correspond to *global minima or maxima* points, *local minima or maxima* points, or *saddle* points. Global minimum points correspond to the globally smallest cost function value, while other minima are classified as local minima (and similarly for global maxima and local maxima). However, saddle points also satisfy the condition present in equation 2.37, but they differ in that they are points where a surface or function curves upward in one direction (forming a maximum) and curves downward in another direction (forming a minimum) when viewed from different angles, as illustrated in figure 2.10.
The primary goal of gradient-based optimisation is to locate the global minimum or a suitable local minimum. Backpropagation helps to efficiently employ these gradient-based optimisation techniques.
Gradient descent, also known as *steepest descent*, is one of the most widely used optimisation algorithms [126]. Its objective is to minimise a cost function, J, with respect to a set of parameters, represented as θ, by making small adjustments to θ in the direction opposite to the gradient at each iteration [19].
According to this optimiser, the parameters vector θ is iteratively updated according to the equation:

$$\theta_{k+1} = \theta_k - \alpha \nabla J(\theta) \tag{2.38}$$

where,

- θ_{k+1} is the updated parameter vector at iteration $k+1$
- θ_k is the current parameter vector at iteration k
- α is the *learning rate*, a hyperparameter represented by a positive scalar determining the size of the step we take to reach a minimum [126]
- $\nabla J(\theta)$ is the gradient of the cost function J with respect to the parameters θ

The learning rate, α, is crucial for determining the size of each step in the iteration process. A large α can cause the optimisation algorithm to overshoot, potentially missing the local minimum. Conversely, a small α can lead to slow convergence, prolonging the optimisation process. These effects are illustrated in figures 2.11 and 2.12.
The key principle behind this approach is that, given the gradient represents the direction of the steepest slope of a function mathematically, the opposite direction of the gradient at a specific point is applied to decrease the cost function as quickly as possible towards a minimum. Convergence is typically achieved when the gradient of the cost function becomes negligible, indicating that the algorithm has reached a minimum.

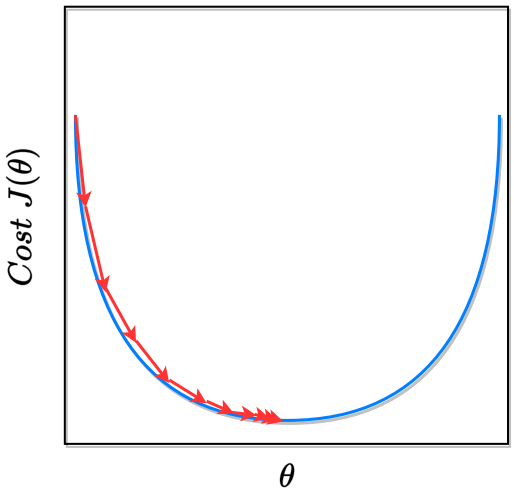


Figure 2.11: A low learning rate produces numerous little gradient steps, which eventually lead to the local or global minimum.

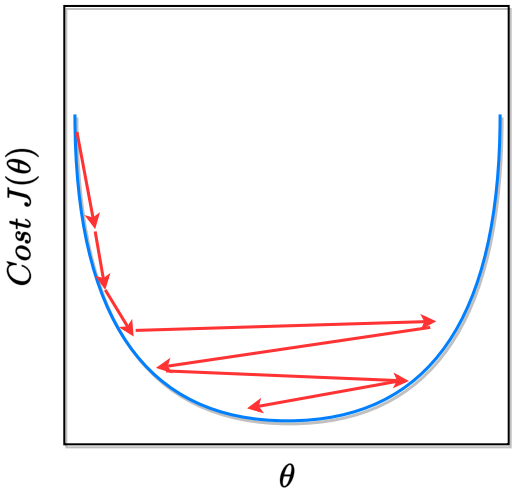


Figure 2.12: A greater learning rate results in longer gradient steps, which may cause the local or global minimum to be missed.

2.6.3 Stochastic Gradient Descent

Stochastic gradient descent is a gradient-based optimisation technique that evaluates either a single random training example or, more commonly, a subset (or mini-batch) of random training examples of size P. Mini-batches typically consist of 32-512 instances, $P \in \mathbb{Z} \cap [32, 512]$, as cited in [66]. Unlike other gradient-based optimisers that calculate the gradient of the cost function based on the entire training set, which can be computationally costly for very large datasets, stochastic gradient descent calculates the gradient of the cost function with respect to a single training example, proceeding in the direction of the negative gradient, as per demonstrated in equation 2.39.

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta J(\theta; \mathbf{x}_i; \mathbf{y}_i) \tag{2.39}$$

where,

- θ_{k+1} represents the updated parameters at the $(k+1)^{th}$ iteration.
- θ_k represents the parameters at the k^{th} iteration.
- α is the learning rate.
- $\nabla_\theta J(\theta; \mathbf{x}_i; \mathbf{y}_i)$ is the gradient of the cost function J with respect to the parameters θ evaluated at each training example $\mathbf{x}_i$ and label $\mathbf{y}_i$.

As a result, while other gradient-based optimisers become slower with larger training sets, stochastic gradient descent is significantly faster, especially for large datasets, as it analyzes

each instance separately. This allows it to update the model's parameters more quickly and converge to the global or local minimum in less time (figure 2.13).

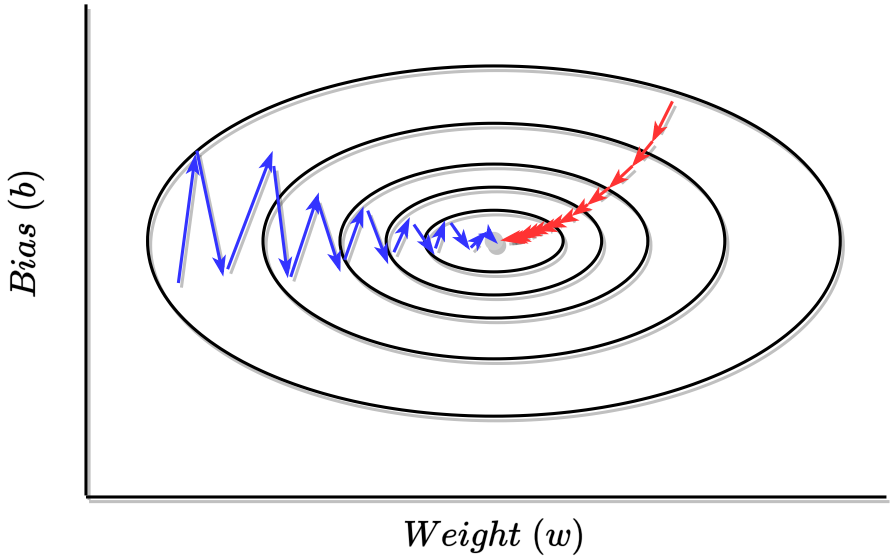


Figure 2.13: Figure illustrating the comparison of the convergence towards a global or local minimum between the mini-batch stochastic gradient descent, represented with the blue arrows, and the 'traditional' gradient descent optimiser, represented by the red arrows.

2.6.4 Momentum

Momentum is an optimisation technique that can significantly accelerate the convergence of gradient-based optimisation algorithms, like stochastic gradient descent. Although, momentum is not an optimiser itself but a technique utilised to enhance some optimisers, its inclusion is relevant since it will be important for fully understanding how the *Adam* optimiser works, an optimiser that will be described in a later sub-section.

Figure 2.13 illustrates the difference in data updating between traditional gradient descent and stochastic gradient descent. For the sake of brevity, the mention of stochastic gradient descent in this thesis refers to the mini-batch implementation. As noted by Sutton et al. [138], a common drawback of stochastic gradient descent is its tendency to oscillate on multidimensional error surfaces, especially when one or more dimensions have disproportionate slopes. This oscillation can make it challenging to find a suitable local minimum, as represented by the zig-zags in blue in figure 2.13.

The primary purpose of the momentum technique is to help optimize the search for the optimal cost value, especially in scenarios where the error surface has sharp curves, typically near local optima [126]. This adjustment helps the algorithm maintain a more consistent and direct path

towards the optimal cost value. In simple terms, the momentum technique can be compared to the behaviour of a rolling ball. When the ball rolls down a hill, it gathers momentum, allowing it to overcome small bumps and irregularities on the surface. Similarly, in optimisation problems, the momentum technique helps the optimiser overcome oscillations and sharp curves in the error surface, making the path towards the minimum smoother and more efficient. This concept is especially important in scenarios where the error surface is complex and multidimensional. In such cases, the momentum technique helps the optimiser maintain a more consistent and direct path towards the optimal solution, ultimately leading to faster convergence and improved performance.

In understanding how momentum functions, it is necessary to first comprehend the mathematical principles that form the foundation of momentum, notably *Exponentially Weighted Average* (EWA).

In EWA we begin with a time series dataset in which each point represents a value at a specific time. The average is then calculated at each time step as we move through the data points. This average is calculated in a way where a higher weightage and significance is given to the most recent points present in the time step in focus. The following equation demonstrates how this is done:

In EWA, a time series dataset is used, where each instance represents a value at a specific time. An average is then calculated at each time step as the data instances are traversed. This average is calculated in a manner that gives higher weightage (referring to the relative importance or influence of a certain variable in a calculation or model) and significance to the most recent points present in the time step under consideration, as per demonstrated in equation 2.40.

$$V_k = \beta \times V_{k-1} + (1-\beta) \times \theta_k \tag{2.40}$$

where,

V_k represents the average at the time step k, with $k \in \mathbb{N}_0$. It is computed by multiplying a hyperparameter β (where $\beta \in [0,1]$) with the average of the previous time step, V_{k-1}, signifying the trend weightage. This term captures the contribution of past averages or other values in the time series to the current average. Additionally, the complement of the hyperparameter, $(1-\beta)$, is multiplied by the parameter(s) of the network that are of interest at time step k, θ_k, representing the current value weightage.

As the time domain is traversed, the following equations can be considered:

$$V_0 = 0 \tag{2.41}$$

$$V_1 = \beta \times V_0 + (1-\beta) \times \theta_1 \tag{2.42}$$

$$V_2 = \beta \times V_1 + (1-\beta) \times \theta_2 \tag{2.43}$$

$$V_3 = \beta \times V_2 + (1-\beta) \times \theta_3 \tag{2.44}$$

A single equation can be derived that generalizes all the above equations, where:

$$V_k = (1-\beta)(\theta_k + \beta \times \theta_{k-1} + \beta^2 \times \theta_{k-2} + \ldots + \beta^{k-1} \times \theta_1) \tag{2.45}$$

or the more simplified formula:

$$V_k = (1-\beta) \sum_{i=1}^{k} \beta^{k-i} \times \theta_i \tag{2.46}$$

where,
V_k represents the EWA at the k^{th} time step, θ_i symbolizes the parameter(s) of the network at the i^{th} time step, and β^{k-i} serves as the weighting factor for the parameter at the i^{th} time step. The hyperparameter β, a constant value ranging from 0 to 1, governs the rate at which previous observations diminish in influence as they recede further into the past. The term $(1-\beta)$ acts as a constant multiplier, ensuring that the sum of all the weighting factors equals 1. This adjustment is crucial for maintaining the overall scale of the Exponentially Weighted Average as the number of time steps increases, which helps in preserving the integrity and relevance of the calculated averages over time.
Through these generalised formulas it is easy to observe how relatively lower weightage is given to older data points and higher to newer ones. If $k >> \beta$, then $\beta^k \approx 0$, indicating older data points begin to become negligible and the newest data points take precedence.
By giving a higher weight to β, the prior average V_{k-1} will be given a bigger weightage than the current point θ_k. This signifies that the model values the trend of the data more than the current instance, therefore it will respond to changes from the current values more slowly. This is good to ignore outliers.
Now, the EWA will be utilized to implement *momentum*. The gradient descent is determined through the formula presented in equation 2.38, which is the same as:

$$\theta_1 = \theta_0 - \alpha \frac{\partial J}{\partial \theta} \tag{2.47}$$

The variation of the cost J concerning the parameters w and b individually can be calculated by:

$$w_1 = w_0 - \alpha \frac{\partial J}{\partial w} \tag{2.48}$$

$$b_1 = b_0 - \alpha \frac{\partial J}{\partial b} \tag{2.49}$$

In order to calculate the momentum, the following substitutions will be made for $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$:

$$w_1 = w_0 - \alpha V_{dw} \tag{2.50}$$

$$b_1 = b_0 - \alpha V_{db} \tag{2.51}$$

Where V_{dw} and V_{db} are the EWA of the weights and biases.
If this process were repeated for k iterations, the resulting calculation would be:

$$V_{dw_k} = \beta \times V_{dw_{k-1}} + (1-\beta) \times dw \tag{2.52}$$

$$V_{db_k} = \beta \times V_{db_{k-1}} + (1-\beta) \times db \tag{2.53}$$

Where, $dw = \frac{\partial J}{\partial w}$, and $db = \frac{\partial J}{\partial b}$, and $\beta = 0.9$ as standard value [46].

Thus, the updation for each parameter is defined by:

$$w_k = w_{k-1} - \alpha V_{dw_k} \tag{2.54}$$

$$b_k = b_{k-1} - \alpha V_{db_k} \tag{2.55}$$

Since the EWA is being integrated into the momentum calculation, and based on our earlier observation that higher values of β allocate more weightage to the trend rather than the current point, resulting in a smoother average line, the parameters w and b will experience less noise in the data and fewer oscillations. Consequently, the overall cost $V_{d\theta}$ will converge more rapidly towards the optimal value (as shown in figure 2.14).

2.6.5 RMSprop

RMSprop (Root Mean Square propagation) is an unpublished optimisation technique described in a Coursera lecture by Geoff Hinton [126]. The RMSprop method accelerates the convergence of the training data into the optimum value by changing the gradient accumulation into an EWA that excludes history from the oldest data instances.
In subsection 2.6.2, the weight updation in gradient descent was observed, which is represented by equation 2.38. Additionally, equations 2.48 and 2.49 can be employed to analyze the fluctuations in the cost J based on the parameters w and b individually.
The substitution for $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$ for the calculation of RMSprop can be done by:

$$w_1 = w_0 - \alpha \frac{dw}{\sqrt{S_{dw_k} + \varepsilon}}, \qquad S_{dw_k} = \beta \times S_{dw_{k-1}} + (1-\beta) \times (dw)^2 \tag{2.56}$$

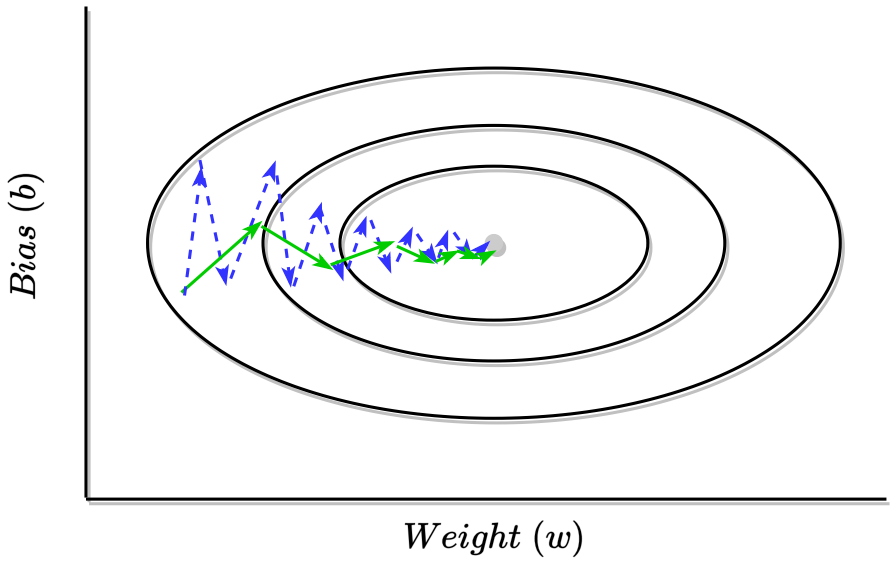


Figure 2.14: This image represents the 2D contour image of a 3D Cost function with parameters b and w, where each black line corresponds to a certain cost value. Comparison between the stochastic gradient descent showed in blue, with the stochastic gradient descent with momentum showed in green. By smoothing the oscillations, reducing the vertical movement in the zigzag and improving the horizontal movement, the stochastic gradient descent with momentum manages to find the optimal value of the cost function faster.

$$b_1 = b_0 - \alpha \frac{db}{\sqrt{S_{db_k} + \varepsilon}}, \qquad S_{db_k} = \beta \times S_{db_{k-1}} + (1-\beta) \times (db)^2 \tag{2.57}$$

where,

- $dw = \frac{\partial J}{\partial w}$ and $db = \frac{\partial J}{\partial b}$.
- w_1 and b_1 are the updated value of weight and a bias parameter, respectively, after one iteration of the RMSprop algorithm.
- w_0 and b_0 are the current value of the weight and bias parameter, respectively.
- α is the learning rate.
- $\sqrt{S_{dw_k} + \varepsilon}$ and $\sqrt{S_{db_k} + \varepsilon}$ are the gradient of the cost function with respect to the weight and bias parameter, respectively, divided by the square root of the moving average of the squared gradients S_{dw_k}, and S_{db_k} plus a small constant ε (to avoid division by zero).

- S_{dw_k} and S_{db_k} are the moving average of the squared gradients of the cost function with respect to the weight and bias parameter, respectively, up to iteration k.
- β is a decay rate parameter between 0 and 1 that controls the exponential decay of the moving average of the squared gradients.
- $(dw)^2$ and $(db)^2$ are the squared gradient of the cost function with respect to the weight and bias parameter, respectively.

According to Goodfellow [46], $\varepsilon = 10^{-8}$ and β=0.999 as standard values.
S_{dw_k} and S_{db_k} are given by the EWA of $(dw)^2$ and $(db)^2$ respectively.

Figures 2.13 and 2.14 clearly demonstrate that the oscillations in stochastic gradient descent occur more prominently in the vertical direction (associated with bias parameter values) rather than the horizontal direction (associated with weight parameter values). Looking at equations 2.56 and 2.57, we can infer that when there is a significant vertical movement, the value of db is likely to increase, consequently leading to an increase in the value of S_{dbk}. This, however, results in a smaller division quantity when db is divided by $\sqrt{Sdbk + \varepsilon}$, thereby causing a reduced overall vertical oscillation. Conversely, if the horizontal movement of parameter w is smaller, the overall quantity of $\sqrt{Sdw_k + \varepsilon}$ will be larger, thereby amplifying the horizontal movement during the update process.
Reducing the vertical movement while increasing the horizontal movement implies that the net movement of the optimisation process will be more direct and cover larger distances towards the optimum value. Consequently, the model will train more rapidly. Although the RMSprop optimisation algorithm enhances the overall efficiency of the model, if the values of S_{dw} or S_{db} approach zero, the values of w and b may overshoot, causing the model to pass the optimum value. To prevent this issue, a very small value, ε, is added to the denominator of the update equations to ensure that the division operation is not affected by near-zero values, thus maintaining a stable optimisation process.

2.6.6 Adam

The *Adam* optimisation algorithm (derived from the phrase "adaptive moment estimation") [68] is an extension of the stochastic gradient descent, for updating the model parameters during the training process. While the stochastic gradient descent maintains a single unchangeable learning rate for all weight updates during training, the Adam optimiser is the result of the combination of *RMSprop* (Root Mean Square Propagation), which adapts the learning rate to the dimension of the gradient, and *momentum*, which helps accelerate the convergence of the model to the optimal solution. Currently Adam is one of the most powerful and widely used optimisation algorithms in deep learning.

The weight updation in Adam is made by combining both the $V_{d\theta}$ of the momentum, and the term $\sqrt{S_{d\theta} + \varepsilon}$ of the RMSprop.

$$w_1 = w_0 - \alpha \times \frac{V_{dw}}{\sqrt{S_{dw} + \varepsilon}} \tag{2.58}$$

$$b_1 = b_0 - \alpha \times \frac{V_{db}}{\sqrt{S_{db} + \varepsilon}} \tag{2.59}$$

Since both the momentum and the RMSprop have hyperparameters β but they take different values, normally the β of the momentum is labelled as β_1 and the β of the RMSprop is labelled as β_2.
Where $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$ (standard values) [46].
By combining both the momentum technique and the RMSprop optimisation algorithm, Adam possesses the advantages of both. Thus, the resultant is better than either of them alone. This way, by utilizing the Adam optimiser, a smoother path can be defined, leading to a reduction in the time required to train the model.

2.6.7 Dropout

Dropout is an algorithm utilised for training neural networks where its objective is to prevent overfitting [8]. *Overfitting* occurs when a model learns to perform well on the training data but fails to generalise to new, unseen data. Dropout helps to address this issue by randomly dropping a proportion of the neurons in a layer during training. This means that at each training step, some neurons are randomly ignored, along with their connections, which forces the network to learn a more robust representation of the data. By randomly dropping neurons and their connections during each training iteration, dropout effectively adds noise to the network [133], which can prevent it from getting stuck in local minima and help it explore a larger range of potential solutions.
Dropout is typically applied during the training phase and is turned off (all neurons are used) during testing. This means that the predictions made by the model during the testing phase are not affected by dropout.

Chapter 3

Convolutional Neural Networks

3.1 Introduction to Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a category of deep learning models introduced by LeCun et al. in 1998, inspired by the visual cortex of animals [77]. Although their first implementation consisted in addressing image pattern recognition problems within the field of *computer vision* – branch of artificial intelligence that allows systems and computers to interpret digital images, videos, and other visual inputs to provide meaningful information – CNNs have since undergone significant evolution, finding applications across diverse domains. These applications encompass object tracking [35], pose estimation [142], text detection and recognition [55], visual saliency detection [159], action recognition [28], scene labeling [36], among others [101].

Multilayer networks trained with gradient descent offer exceptional capabilities for comprehending complex, high-dimensional, and nonlinear mappings from large sample sets, making them promising candidates for image recognition tasks [73]. The *Neocognitron model*, proposed by Kunihiko Fukushima in 1980 [42], laid the groundwork for many of the ideas later incorporated into modern CNN architectures. Through the incorporation of gradient descent algorithms into the Neocognitron model, Yann LeCun developed one classic model for pattern recognition, known as the *LeNet model*, in 1998 [77]. This model utilised the modified National Institute of Standards and Technology (MNIST) database, which is a structured collection of data comprised of several datasets. The MNIST database comprises a vast collection of handwritten digits ranging from 0 to 9, encompassing a training set of 60,000 examples and a test set of 10,000 examples. This database is derived from a larger collection, the NIST Special Database 3 (featuring digits written by United States Census Bureau employees) and Special Database 1 (including digits penned by high school students) [77]. The database consists of binary images of handwritten digits, where each pixel can only have one of two possible values, 0 or 1, corresponding to "black" and "white", as illustrated in figure 3.1. The resulting images are 28×28 pixels in size.

The LeNet model was designed to process 28×28 pixel images of handwritten single digits as

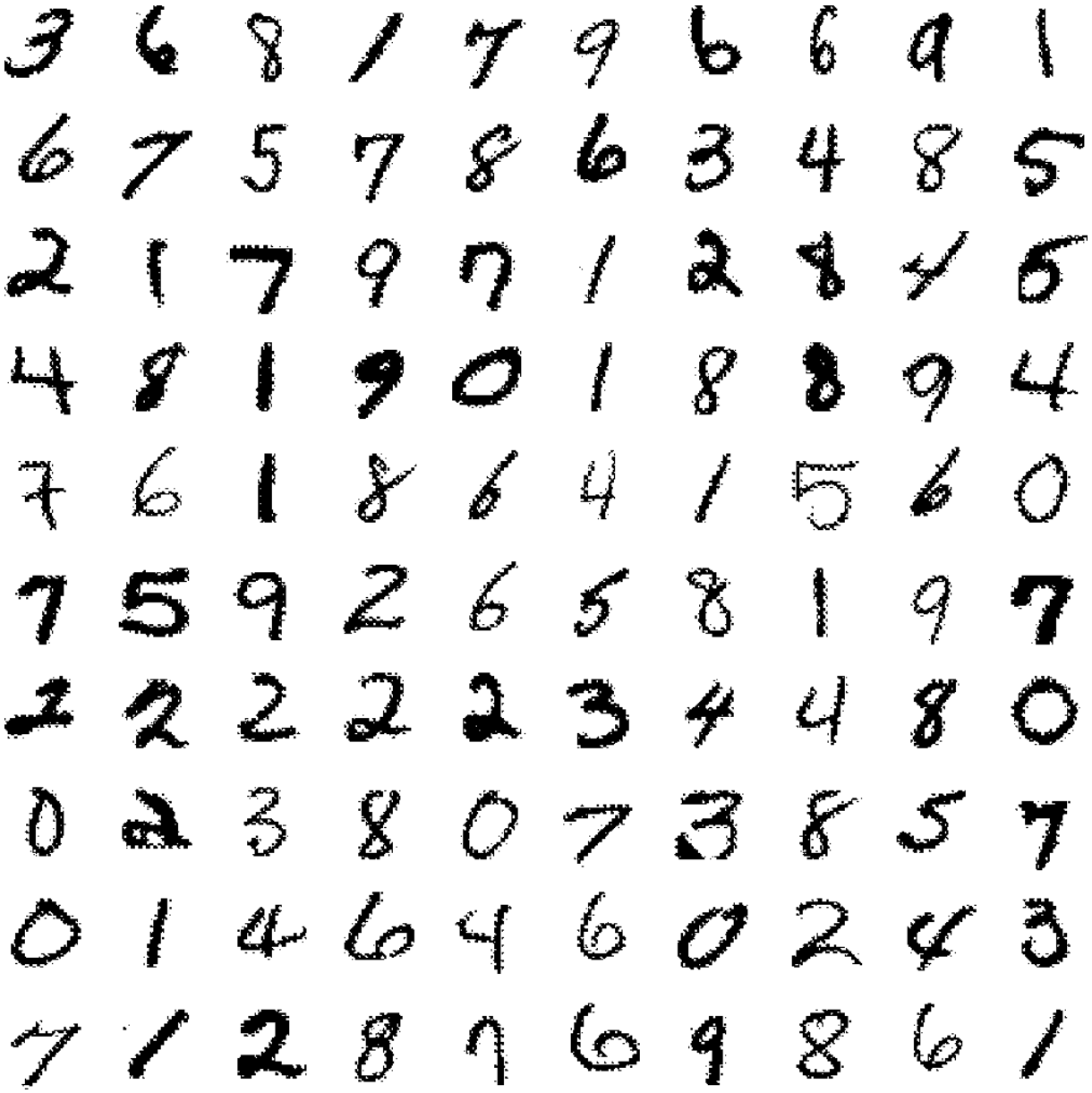


Figure 3.1: Size-normalized examples from the MNIST database, imaged adapted from [77].

input by passing them through a CNN. Further details on the steps involved in this process will be provided later. After processing the 28×28 pixel images of handwritten digits, each image is changed into a single vector that has a dimension of 784. This dimension number results from the multiplication, $28 \times 28 = 784$, which represents the pixels in the image. This vector, with specific features of the images, is the input data that the model focuses on identifying and using to classify or categorise the input features. In the case of the MNIST database, the features of

an example might include the shapes of individual strokes or loops that form the digits. The primary objective is for the model to learn how to recognise these features and to correctly predict the digit represented by the inputted number (ranging from 0 to 9) [77].

When addressing tasks of this nature, conventional fully connected networks – artificial neural networks where each neuron in one layer is connected to every neuron in the next layer– face inherent challenges, particularly in handling the complexity and spatial relationships present in image data. This complexity entails understanding the relative positions, sizes, orientations, and distances between different elements within an image, reinforcing the importance of utilising specialised architectures like LeNet, and CNNs in general, for optimal performance in such tasks. This preference is due to the fact that fully connected networks, including feedforwad neural networks, inherently possess a vast number of parameters. To exemplify this, let's considered a 10×10 pixel gray scale image. In a fully connected network, each pixel is connected to every neuron (figure 3.2). Just in the initial hidden layer, with merely 10 neurons, 110 parameters, including biases, are required (10 pixels × 10 neurons + 10 bias terms). If the image dimensions were increased to 100×100 pixels, which is still a very small image, the number of parameters would rise significantly to 10,100 in the first layer alone. Consequently, fully connected networks, demand a large training set to effectively learn these parameters and capture the intricate patterns present in the data. Furthermore, they are not inherently invariant to translations or local distortions in the input. In other words, if an image or input undergoes changes such as shifting its position (translation) or distorting its local structure, like warping, stretching, or deforming the pixels within a small area of the image, fully connected networks struggle to recognize or classify the input correctly. Handwriting, as in the MNIST example, implies a lot of variability and different writing styles, which produce changes in the position of distinctive features in input objects. CNNs overcome this limitation by restricting the scope of analysis of hidden neurons to local regions, the receptive fields, in order to successfully detect and learn relationships between nearby pixels or features within the input data (capture local correlations) [67]. This means that instead of considering the entire input data at once, CNNs analyse small portions of the input data at a time, allowing them to capture local correlations more effectively, as illustrated in figure 3.2.

Sparse interactions, *parameter sharing*, and *equivariant representations* are three key concepts that CNNs use to improve on the computational capabilities of generic artificial neural networks [46].

CNNs, unlike classic neural networks, have *sparse interactions* because they use smaller *kernels/filters* – small matrices or set of parameters – that focus on significant features, such as edges in an image, within a subset of the data, resulting in partial rather than full connections between layers (Fig. 3.3). By limiting the number of connections to a smaller value, the runtime for matrix multiplication is significantly reduced. This sparsity decreases the number of parameters, improves memory efficiency, and speeds up the computation. Such improvements

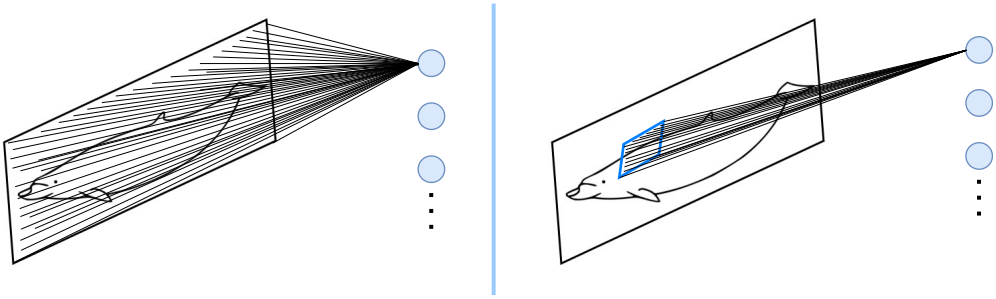


Figure 3.2: Figure depicting input images of a beaked whale. On the left side, the connections per neuron (blue circles) in a fully connected neural network are illustrated, represented by the black lines. On the right side, the connections per neuron are concentrated in a receptive field, which is represented by the blue square. This field enables the capture of local correlations, a characteristic feature of CNNs.

to efficiency are especially relevant in practical applications.

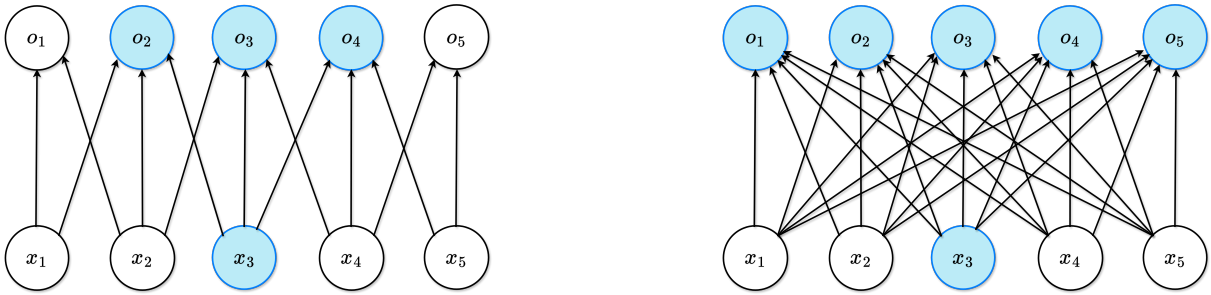


Figure 3.3: This figure depicts a comparison between an artificial neural network with sparse connectivity (left image) and a fully connected artificial neural network (right image). One input neuron, x_3, is highlighted as well as the output neurons, o_i with $i \in \{1,2,3,4,5\}$ affected by the input neuron. When o is produced by applying a kernel with a width of 3, only three outputs are affected by x (left image). However, in the full connectivity case, when o is formed through matrix multiplication, the sparse connectivity characteristic of the previous method is lost, resulting in all outputs being influenced by x_3. Image adapted from [46].

Parameter sharing is another crucial trait of CNNs that allows feature detectors to be employed across different regions of an image, optimising their utility in numerous locations. In a fully connected artificial neural network, the computation of the output involves multiplying each parameter in the weight matrix with a single feature input, after which it is not used again. However, in CNNs, the kernel weights are applied to the entire input. In other words, the weights are reused at every location in the image where the kernel is applied. For example, for detecting vertical edges in an image, instead of having a separate kernel for each location in the

input image, CNNs use the same kernel (or set of weights) to detect vertical edges throughout the entire input image. This means that every part of the image is analysed using the exact same kernel. By sharing parameters in this way, each output neuron can detect the same features, such as vertical edges, across different parts of the image.

The third key feature employed by CNNs to enhance the computational capabilities of generic artificial neural networks is their ability to exhibit *equivariant representations*, also known as equivariance to translation. Equivariant representations entail that if there are certain transformations or symmetries in the input data, these symmetries are preserved in the representations learnt by the network, meaning that alterations in the input lead to corresponding alterations in the output. This property allows the network to grasp the relationships between different parts of an image or any spatial data, relationships denominated as spatial connections, when performing tasks like image processing. For example, in a picture of a face, spatial connections involve how the eyes, nose, and mouth are positioned in relation to each other, and equivariance allows the network to capture these relationships efficiently using sparse connectivity.

Now that the key features that grant CNNs their ability to effectively process and recognise patterns in structured data, particularly in tasks involving spatial relationships such as image processing and computer vision, have been clarified, it is necessary to delve into their composition. CNNs are composed of three distinctive types of layers, as illustrated in figure 3.4, the *convolutional layers*, *pooling layers*, and *fully-connected layers*.

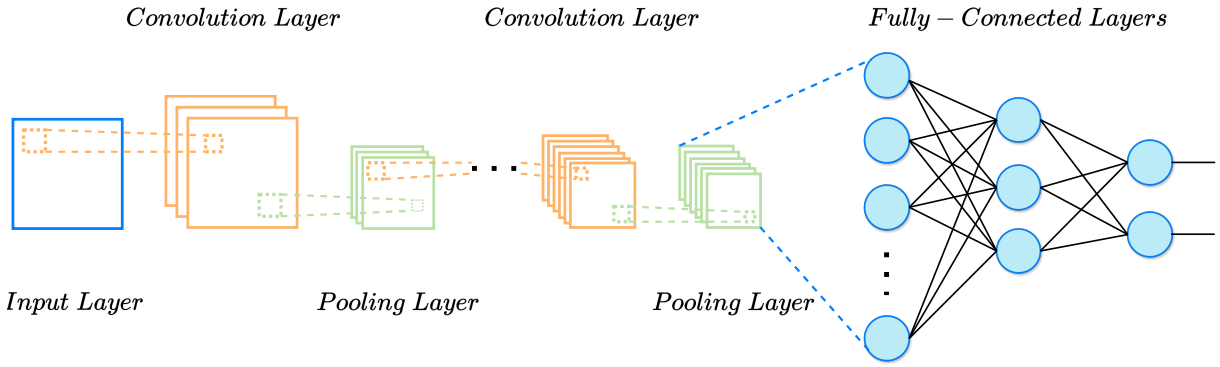


Figure 3.4: Architecture of a basic CNN with its different types of layers.

Goodfellow et al. highligths the absence of a universally optimum CNN architecture [46]. The set of standards is constantly changing as CNNs grow in popularity and more research is dedicated towards them. Prominent models such as *AlexNet*, *VGG-16*, and *GoogLeNet* are just a few examples of this constant search for improvement [31]. In the subsequent sections the core operational layers of CNN architectures will be looked into, explaining their function as generic building blocks for the architecture. It is worth noting that detailed exploration of fully connected layers will be omitted here, as their process parallels the one described in Chapter 2.

3.2 Convolution

CNNs get their name from convolutional layers. These layers conduct an operation known as '*convolution*', which is unique to them. Convolution, in its most generic form, is an operation of two functions that outputs a function corresponding to the modification of the two inputs, and can be defined by:

$$F = x * w \tag{3.1}$$

In the equation 3.1, F is the function outputted from the convolution operation between x and w, which is represented by the asterisk symbol '$*$'. Convolutional networks utilise the word "*input*" to refer to the first argument of the convolution and "*kernel*", or *filter*, to refer to the second argument, in this case x and w respectively. The output of this procedure, F, is known as the *feature map* or *activation map*.

A convolutional operation can have several dimensions, the most frequent among them being two dimensional, 2D, and three dimensional, 3D. The spatial dimensions of the input, such as *width* and *height*, are used in a two dimensional convolutional procedure. The variable *depth*, or *number of channels*, is included while dealing with three dimensional convolutional procedures.

When working with grey-scale pictures, for example, the input only has one channel, thus a two dimensional convolutional procedure is applied. However, by working with ***RGB*** images, meaning coloured images, the input contains three channels, one for each colour (Red, Green, and Blue), therefore we are in a three dimensional convolutional procedure scenario.

In discrete convolutions, which are utilized for processing discrete data, a kernel is employed to capture specific patterns or features within an input matrix, or grid of values. This kernel, typically a small matrix of weights, is systematically moved across the input data. The sliding of the kernel initiates horizontally, traversing all the columns of each row within the input matrix. Upon reaching the end of a row, the kernel cycles back to the beginning of that row and progresses to the next row below. This iterative process continues until the entire input matrix has been covered.

At each position of the kernel on the input, a dot product is calculated between the kernel weights and the corresponding values in the input data. This process involves multiplying each element of the kernel by its corresponding element in the input, followed by summing these products. This mathematical operation is defined as a *convolution*. Ultimately, this computation results in a single value, as per illustrated in figure 3.5.

By sliding the kernel across the entire input, the convolution operation calculates a new value for each position. These values collectively form the *feature map*, which represents the output of the convolution operation. This is illustrated in the following equation:

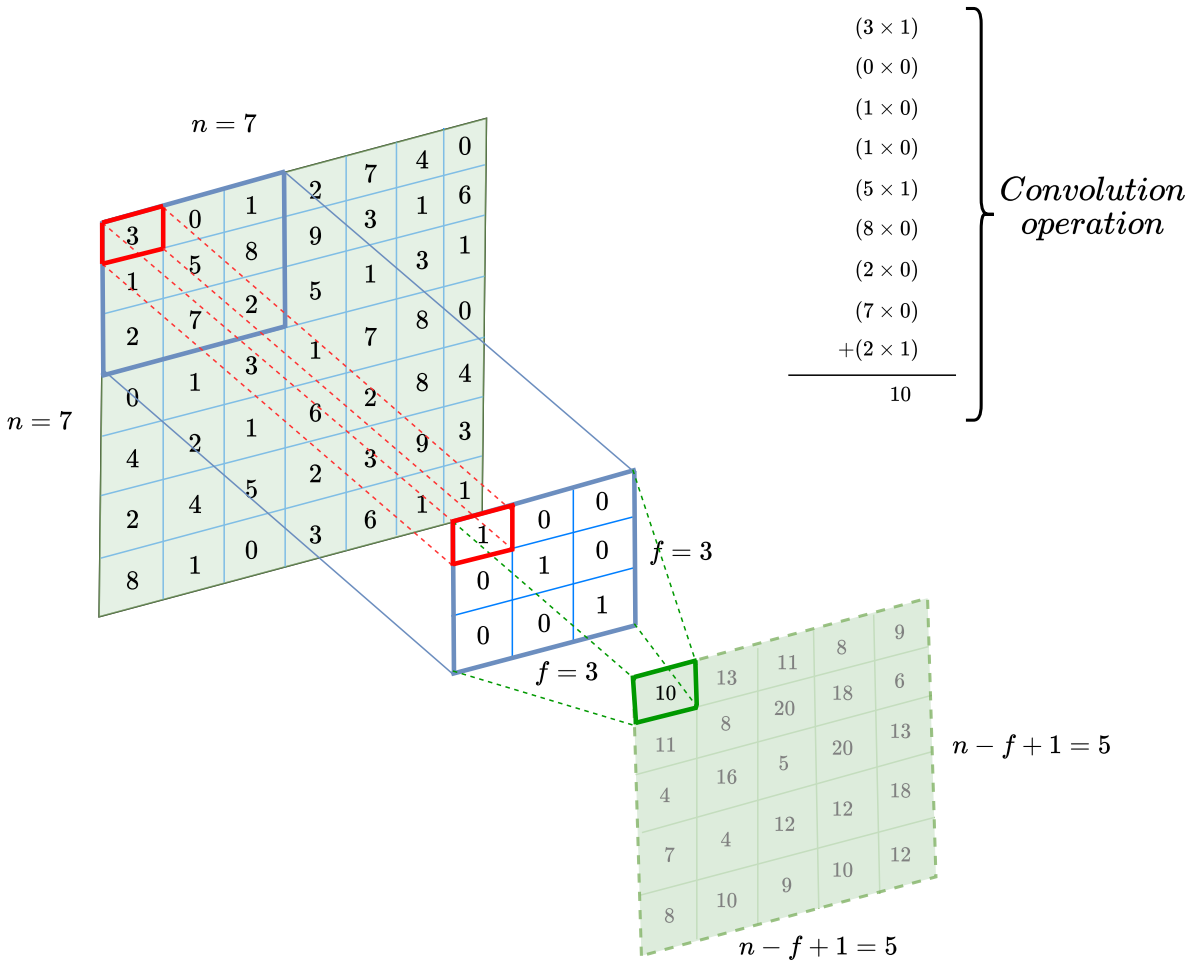


Figure 3.5: Illustration of the convolution mathematical operation: A depiction showing an input matrix on the left, with a kernel applied to it, in the centre, resulting in the generation of a feature map, on the right.

$$F(i,j) = (X * W)(i,j) = \sum_m \sum_n X(m,n) W(i-m, j-n) \tag{3.2}$$

Equation 3.2 represents the mathematical definition of the convolution operation between an input matrix X and a kernel W, resulting in a feature map F, where:

- $F(i,j)$ represents the value at position (i,j) in the resulting feature map F;
- $(X * W)(i,j)$ represents the convolution operation between the input matrix X and kernel W at position (i,j);
- $\sum_m \sum_n$ represents the summation over all possible values of indices m and n;

- $X(m,n)$ and $W(i-m, j-n)$ represent the values of the input matrix X and the kernel W at positions (m,n) and $(i-m, j-n)$, respectively.

Each feature map has its own width, height, and depth. The number of kernels dictates the depth of the feature map, given that each kernel applied to the input data produces a single channel in the feature map. Assuming the input is a two-dimensional matrix $n \times n$ and the kernel is a two-dimensional matrix $f \times f$, where $n > f$, the resultant feature map is a two-dimensional matrix of dimensions $n-f+1 \times n-f+1$. Note, that in this scenario the symbol $\times$ does not symbolise a multiplication but a separation between the rows and columns of a matrix.

3.3 Padding

Padding is a common strategy for mitigating resolution loss during convolution.
As previously stated, if we take a $n \times n$ input picture and convolve it with a $f \times f$ kernel, the output feature map will be $n-f+1 \times n-f+1$ in size. This is because the number of feasible spots to fit an $f \times f$ kernel on a $n \times n$ input matrix, where $n > f$, is only $n-f+1 \times n-f+1$. Nonetheless, there are two drawbacks to this. If the feature map decreases each time a convolutional operator is used, this indicates that the input picture can only be convolved a limited number of times before it becomes excessively small. Another key consideration is that when the kernel convolves in the input picture, the pixels on the corners/edges of the input matrix are less incorporated into the convolution operations than the other pixel values. This implies that most of the information around the margins of the input image is discarded and is not properly represented on the feature map.
Padding is a solution proposed to face these difficulties of output/feature map reduction and discarding information from the image's edges. *Padding* is a technique where additional elements are added around the boundaries of an input before applying the convolutional operation. This technique entails adding to the input data new rows and columns of elements, which are often filled with zeros. In literature there are typically two types of padding used in convolutional operations, *valid padding*, and *same padding*, as per illustrated in figure 3.6.
Valid padding entails the absence of padding at all, the kernel only convolves where it fully overlaps with the input. This results in the feature map reduction scenario afore mentioned.
Same padding ensures that the dimensions of the feature map remain the same as the input's. It accomplishes this by equally padding the input all around, allowing the kernel to cover all positions. The amount of padding applied to either side of the input, assuming the number of rows is the same as the number of columns, is calculated by:

$$p = \frac{(f-1)}{2}, \tag{3.3}$$

where p corresponds to the number of elements that will be added to the rows or columns of

the input image, and f to the size of the kernel's width or height. The resulting dimensions of the feature map when applying same padding are $n+2p-f+1 \times n+2p-f+1$. Note, that the symbol $\times$ does not symbolise a multiplication but a separation between the rows and columns of a matrix.

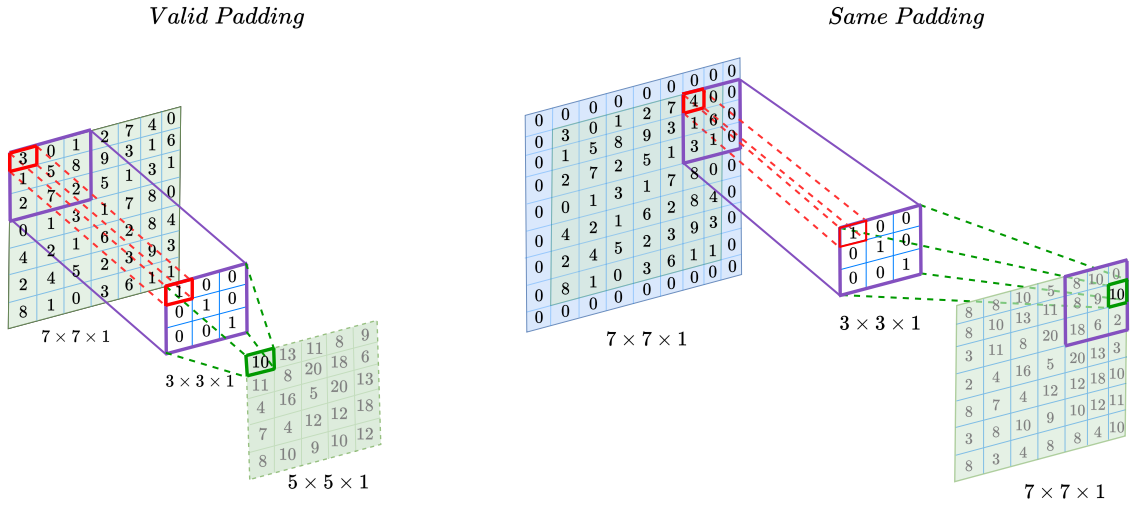


Figure 3.6: This illustration demonstrates two padding techniques. The first, referred to as "valid padding", results in a feature map with reduced dimensions. The second, known as "same padding", involves adding extra elements around the input's boundaries, ensuring that the resultant feature map retains the original dimensions.

3.4 Stride

Strides in CNNs, often simply noted by S, are another piece of the basic building blocks, referring to the number of positions by which the convolutional kernel traverses through the input data. Convolutions are performed by sliding the kernel over the input data and calculating a dot product between the kernel and the corresponding values in the input at every location. The *stride* indicates the distance between successive positions where the kernel is applied. Typically, the most employed stride value is of 1, in order to ensure a more thorough examination of the input due to the overlapping zones between successive applications of the kernel. By raising the stride values, the spatial dimensions of the feature map may be reduced while the computational efficiency is increased. These options are more appealing when a more coarse-grained examination of the input is desired or a down sampling is required.
When considering a convolution with a stride S, assuming the input image corresponds to a matrix of dimensions $n \times n$, and the kernel of dimensions $f \times f$, the resulting featured map is governed by the formula:

$$\frac{(n+2p-f)}{S}+1 \tag{3.4}$$

Thus, the feature map will result in a matrix with dimensions $\frac{(n+2p-f)}{S}+1 \times \frac{(n+2p-f)}{S}+1$. Once again, note that the symbol $\times$ does not symbolise a multiplication but a separation between the rows and columns of a matrix.

In the event that the fraction of $\frac{(n+2p-f)}{S}+1$ does not correspond to an integer, the same is rounded down, or floored, to the nearest integer. In other words, if the kernel does not lie entirely within the input matrix, or the input matrix plus the padding region, that computation is not made and the fraction of $\frac{(n+2p-f)}{S}+1$ is rounded down. Figure 3.7 shows an example demonstrating the application of stride.

3.5 Pooling

Pooling is an essential technique in convolutional neural networks, implemented in layers designed as *pooling layers*, that effectively reduces complexity in subsequent layers through down sampling. By condensing the outputs of convolutional layers, it results in a decreased number of inputs for the following layers. This reduction in parameters not only enhances computational efficiency but also optimises the overall network design. In the realm of image processing, it is comparable to lowering picture resolution. Pooling allows to summarise information from neighboring pixels. It works by calculating a statistical value, like the maximum or average, within a specified region of the input. These calculated values then replace or down-sample the outputs at certain locations [46].

The application of these layers helps reduce the spatial dimensions of the data while preserving the number of kernels in each layer of the network, which is important to maintain its ability to learn and represent features effectively. Pooling has the capacity to help the network become less sensitive to small changes in the position of features, making it more robust and adaptable to variations in the input data. This implies that even if the input is partially translated, the pooled output values are unaffected since pooling focuses more on the presence of features rather than their precise location [46]. This characteristic is notably useful in object identification jobs.

The most commonly used type of pooling layers in CNNs is *max pooling*. While using this technique, the kernel functions as a batch that partitions the input image, only returning the maximum value present in that batch. One of the most common kernel sizes used in max-pooling is 2×2. When pooling, usually the most common stride value applied is also 2. Although utilising a stride of 1 would avoid down-sampling, usually that is not the main goal when applying pooling layers, thus this stride value is not commonly used [5].

There is one other type of pooling that is not used as often called *average pooling*. In this type of pooling, instead of taking the maximum values within each kernel, the average of all values

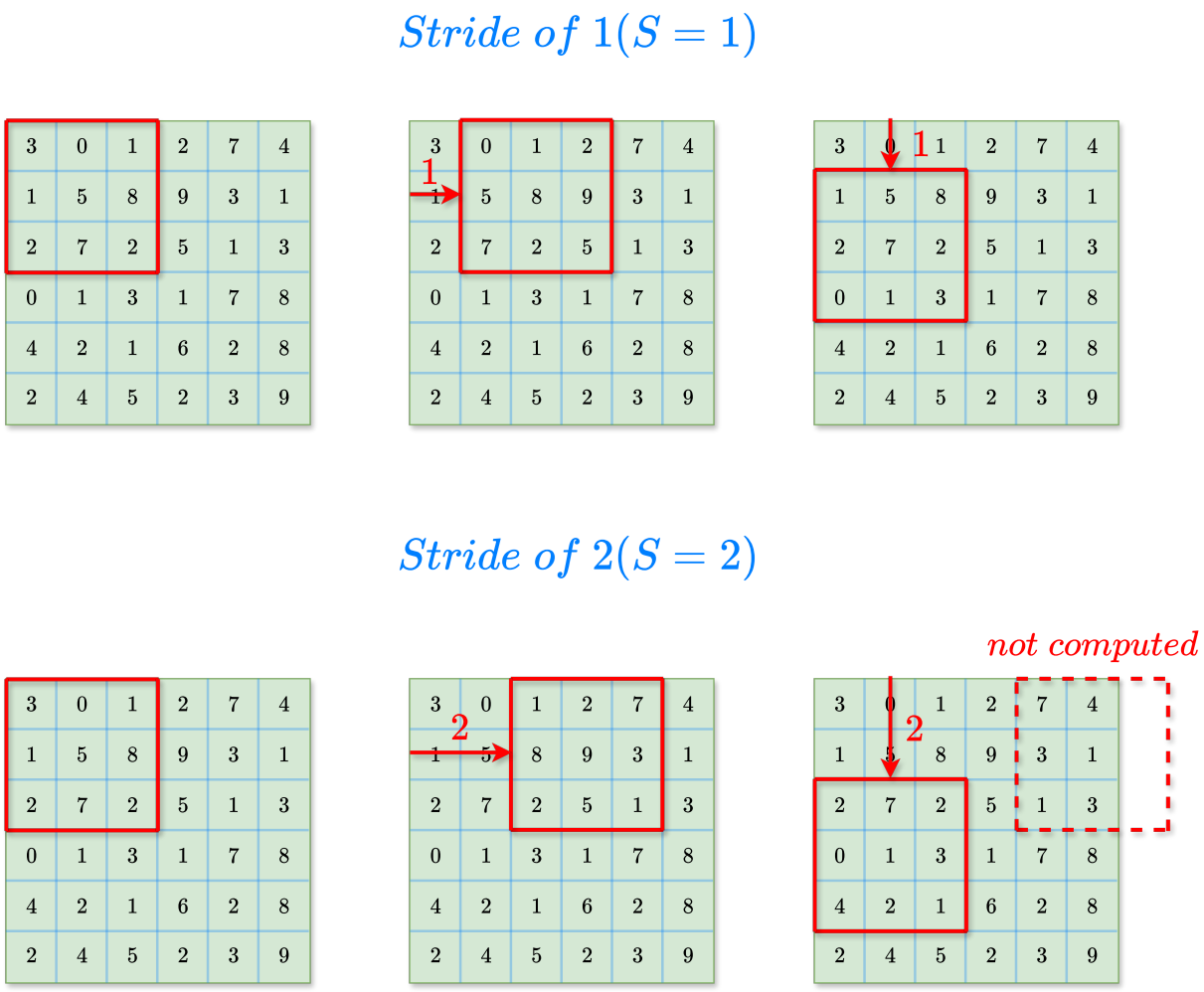


Figure 3.7: This illustration showcases the impact of two different stride values. In the upper section, a stride of 1 is applied, resulting in the kernel shifting one position at a time. The third image demonstrates the kernel moving one position down, which occurs after traversing all columns in the upper row. In the lower section, a stride of 2 is utilized. Here, the kernel shifts two positions at a time. The third image highlights that when the kernel does not fit entirely within the input matrix, the operation is not executed, sliding two positions downward after covering all columns in the upper rows.

encompassed by the kernel is taken instead. Figure 3.8 illustrates an example showing how both types of pooling layers function.

Figure 3.8: This illustration demonstrates the reduction of feature map dimensions following the application of max pooling, on the left side, and of average pooling, on the right side. For these instances, a 2×2 kernel with a stride of 2 was employed. The colour scheme represents the kernel's positions on the input matrix and their corresponding outputs in the feature map.

Chapter 4

Deep Learning for bioacoustics

4.1 Introduction to bioacoustics

Bioacoustics, a scientific discipline merging insights from biology and acoustics [141], delves into the study of sound generation and reception in animals, as well as their communication mechanisms [112]. In particular, marine bioacoustics focuses on the application of acoustic techniques to examine various aspects of marine animals, including their auditory abilities, production of sounds, communication patterns, and foraging behaviour [7].

Sound can be defined as a form of energy that travels through a medium like gas, liquid, or solid in the form of vibrations, creating what is perceived as sound waves. One of the central aims of bioacoustics is to understand the relationship between the sounds that animals produce and the environment in which they are employed, as well as the specific functions that these sounds serve.

Due to the challenging conditions of underwater environments, where limited visibility can hinder visual observations, alternative strategies are necessary for studying marine animals. Instead of relying on visual perception of their behaviour by analysing landscapes, researchers turn to soundscapes, using techniques such as *passive acoustic monitoring* [90], as acoustic energy propagates more efficiently in water than any other form of energy [149]. A *soundscape* encompasses all the sounds generated by the various components of an environment. This consists of three sources, the "geophony", which refers to the non-biological sounds like the wind in the trees and the waves on a beach, the "biophony", the collective sounds of organisms in their natural habitat, and the "anthrophony", the sounds produced by humans [114].

In the early days of bioacoustic research, analogue recorders were used as the main tools for capturing and reproducing sound or data in its analogue form [79]. This meant that the output generated by these devices mirrored continuous variations in a physical quantity, such as voltage, pressure, or light intensity, which corresponded directly to the original data. During this time, researchers had to transport equipment that was big and heavy, as well as its batteries to field

sites. Not only that, but the length of the recordings was often constrained by high tape and battery usage. Consequently, there were few studies that used bioacoustics-based approaches [113].

Later advances in electronics, such as the digitisation of recordings and equipment miniaturisation, which allowed portability, have transformed bioacoustics. Nowadays, digital information serves as a representation of analog data by taking samples at specific points in time and converting these sample values into binary numbers [116].

Advances in microphones throughout the years allowed the recording of sound at a wider range of frequencies, extending beyond the human hearing range (20Hz-20kHz) [122]. Consequently, this made it possible to make new discoveries and deepen the current knowledge of species that communicate outside of this range, such as bats, elephants, whales, among others [41, 16, 61].

Once a sound is digitally recorded, it is crucial for it to follow the *Nyquist rate*. According to this principle, the sampling frequency must be at least twice the maximum frequency component of the signal to ensure that the signal fidelity during digitization is preserved. Following this guarantees that the sound can be transferred to a computer with ease and high quality, where it can be stored, edited, copied, shared, played, processed, and analysed using various software programs, all without degradation [113]. This allows the study of sounds made by animals across different environments, during day and night, throughout the year, and often remotely. When sounds are recorded remotely, the bioacoustic research is termed 'passive', and commonly referred as *passive acoustic monitoring*. The advantage of passive acoustic monitoring studies lies in their non-invasive nature, allowing anyone with a minimal amount of equipment to record animal sounds [84], either by deploying autonomous recording devices throughout habitats or attaching them directly to animals [16, 145]. These technological advancements have transformed the sampling, processing, storage, and accessibility of sound data.

Basic bioacoustic research equipment is now widely available and reasonably priced, whereas high-end sound recorders, powerful laptop computers, and specialised software have broadened fieldwork possibilities. As a result, datasets of animal sounds from various species are now widely used for research. These sound datasets provide a wealth of information that researchers can use to study many aspects of animal communication, behaviour, and ecological dynamics. For example, by attaching animal-worn acoustic tags to whales, researchers were, not only, allowed to study the movement and acoustic behaviour of the tagged animals, but also their behavioural and sound responses to anthropogenic sounds, such as naval sonars [146].

Thus, bioacoustics has grown as an important area at the centre of biology, acoustics, and behavioural sciences, allowing researchers to understand the intricacies of animal life.

4.2 Bioacoustics of beaked whales

Beaked whales (family *Ziphiidae*) are a group of cetacean species which inhabits mainly in deep oceanic waters [10]. Despite there being 21 recognised species in this family, they remain poorly understood due to their pelagic habitat, small group sizes, and cryptic surface behaviour [10]. Most species of beaked whales are observed in groups of less than five individuals, nonetheless solitary individuals are also commonly seen [10]. The combination of having an elusive nature, together with the lack of conspicuous visual cues, such as blows or splashes, makes studying beaked whales very challenging [10]. As a result, little is known about their abundance or population density, with significant gaps in knowledge persisting regarding their geographic distributions [81].

Recently, studies that focused on the characterisation of echolocation signals, which correspond to the process of creating sound waves and analysing the echoes that are reflected off nearby objects in their environment in order to find and identify those objects, produced by several beaked whales' species, demonstrated that these are species-specific [61, 85]. Through the application of non-invasive techniques, such as the attachment of acoustic recording tags, behavioural studies revealed that two species of beaked whales in particular, Cuvier's beaked whale (*Ziphius cavirostris*) and Blainville's beaked whale (*Mesoplodon densirostris*), produce echolocation clicks throughout their foraging dives (dives in search of prey) [144, 26].

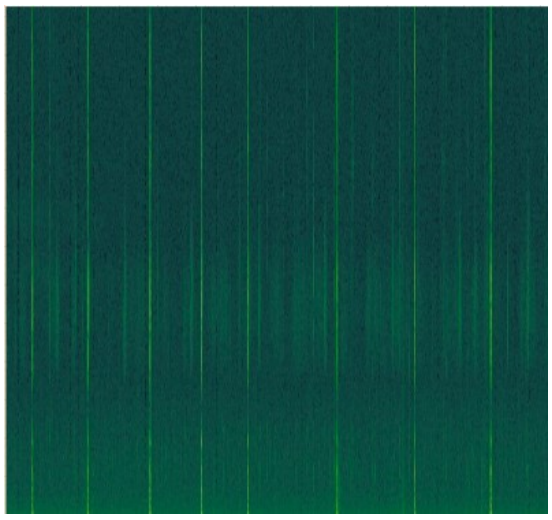
Due to the distinct echolocation clicks produced during foraging dives, passive acoustic survey (application of passive acoustic monitoring techniques to conduct a survey) methods have been suggested as a valuable tool for studying beaked whale density and abundance [89]. Visual survey methods are challenging to apply due to the whales' deep-diving behaviour and difficulty in sighting them at the surface [10].

Blainville's beaked whales are known to predominantly hunt during the descending phase of their foraging dives by echolocating in its deeper parts, primarily focusing on *mesopelagic* (200-1000 metres) and *benthopelagic* fish (fish near the ocean floor), supplemented by cephalopods and crustaceans [151]. During the ascents of foraging dives, they maintain silent, possibly to avoid predators from listening near the surface [6]. These whales possess a unique social structure, living in small, cohesive groups of around four individuals [30], with long periods of association, sometimes for months at a time, which suggest a strong social bond and coordinated activities [30].

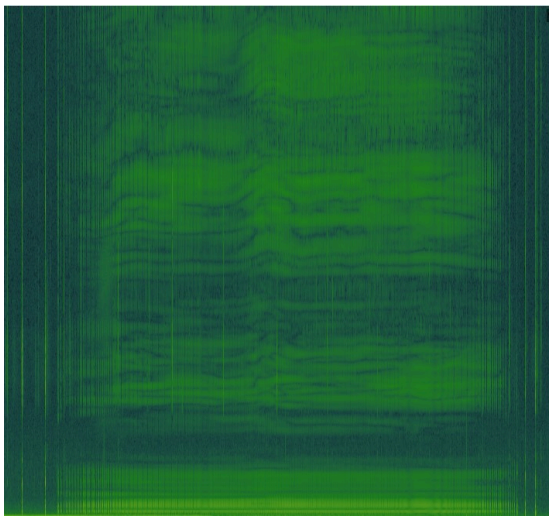
Jones *et al.*, in [63], reported that these elusive marine mammals use broadband (encompass a wide range of frequencies), ultrasonic echolocation signals from 26 to 51 kHz to search for, localise, and approach prey, primarily consisting of mid-water and deep-water fishes and squid. The echolocation clicks they produce are highly directional, ultrasonic transients with duration's around 250 μs, and they have a frequency distribution where the amplitude of the signal remains relatively consistent across a range of frequencies (flat spectrum) from 30 kHz up to the 48 kHz

Nyquist rate of the acoustic sampling.

Their echolocation process during prey capture can be divided into three phases: the search, approach, and terminal phases. Johnson *et al.*, in [58], found that during the search phase, they emit long sequences of regular clicks with interclick intervals (ICI), time between two consecutive clicks, between 300 and 400 ms. These echolocating clicks help the whale navigate through regions with multiple objects or potential prey, as the reflected sound waves provide information on the locations of these objects or potential prey. In the approach phase, the whales continuously ensonify (apply sound waves to an area or an object) a potential prey and receive echoes without changing the production rate of clicks, transitioning to a buzz at the very end. The terminal, or buzz phase, is initiated when the prey is approximately within a body length of the whale (2–5 m). During this phase, the whale starts emitting "buzzes", high-repetition click sequences lasting 2–5 seconds with ICIs of 5–20 ms. The high cadence of these clicks allows the whale to gather an abundance of information on the prey's location, utilising all the reflected sound waves generated by the prey as the echolocation clicks ensonify it. This flow of information allows the whale to make quick movement adjustments in order to capture the prey. Figure 4.1 demonstrates the difference between the ICIs of the regular clicks used during the search and approach phases and the buzzes used during the buzz phase.



(a) Blainville's beaked whale production of regular clicks



(b) Blainville's beaked whale production of a buzz

Figure 4.1: Demonstration of the difference in the interclick intervals during buzz clicks, typical of the buzz phase and regular clicks, typical of the search and approach phases

In addition to their echolocation clicks and buzzes, Soto *et al.*, in [26], identified a fast series of ultrasonic frequency-modulated clicks, which he called "rasps" and harmonically rich short whistles. According to Soto *et al.*, these sounds seem to be associated with the vocal foraging

phase of dives when the whales disperse to forage at depth.

4.3 Acoustics in signal processing

A spectrogram is a visual representation of the frequency content of a signal as it changes over time [44]. To analyse complex signals like sounds and vibrations, spectrograms are often utilised in a variety of disciplines, such as audio signal processing [86], speech analysis [104], and bioacoustic research [29].

4.3.1 DTF and STFT

Before defining the spectrogram, it is important to understand some concepts such as what a signal and a Fourier Transform are.
A signal is a variation in a certain quantity over time. In the audio realm, the changing quantity is air pressure. In order to digitally record this information, it is essential to sample air pressure at different points in time. This captured data is termed a signal's waveform, which can be interpreted, altered, and examined using computer software.
In order to extract useful information from these signal's waveforms a Fourier Transform can be applied.
An audio signal is made of various single-frequency sound waves. When sampling a signal across time, only the amplitudes of the signal at those specific moments in time are captured and recorded. The Fourier transform is a continuous mathematical equation (eq. 4.1) that allows the dismantle of such signals into its individual frequencies and their corresponding amplitudes (fig. 4.2). This transformation essentially changes the signal's representation from time-based to frequency-based, with an outcome known as a spectrum. This transition is possible, due to the fact that any signal can be broken down into an assortment of sine and cosine waves that add up to the original signal.

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-i\omega t} dt \tag{4.1}$$

where $x(t)$ is some inputted continuous time-domain signal as a function of time t, $'e'$ is the base of natural logarithms (Euler's number), $'i' = \sqrt{-1}$ (imaginary unit), $'\omega'$ is the angular frequency in radians per unit time, and $X(\omega)$ is a continuous frequency-domain function represented at angular frequency $'\omega'$. Thus, eq. 4.1 is used to transform an expression of a continuous time-domain function into a continuous frequency-domain, enabling the determination of the frequency content of any signal of interest.
Two of these widely used frequency domain transformations are, namely, the Discrete Fourier Transform (DFT), and Short-Time Fourier Transform (STFT).

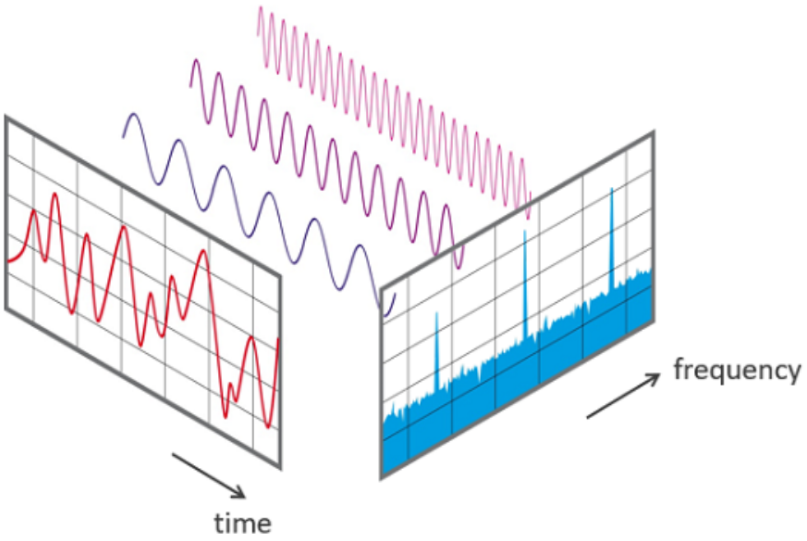


Figure 4.2: Representation of the Fourier Transform being applied to an audio signal, decomposing it in all of its combined signals and converting it to an amplitude spectrum with their corresponding individual frequencies. Image adapted from Google Images.

In practical applications, it is not feasible to perform these calculations over the entire time domain. Consequently, one is limited to a finite-length sampled sequence, typically denoted by L, which is referred to as the window length. However, there still resides the problem of continuing to have a continuous range of frequencies, which makes it challenging for implementation in digital computer applications. To address this, the frequency domain is divided into discrete values through the evaluation of $X(\omega)$ at a finite set ω_k of N equally spaced frequencies spanning a single interval of length 2π. These frequencies are given by $\omega_k = 2\pi k/N$, where k ranges from 0 to $N-1$. This makes the transform into the Discrete Fourier Transform (DFT), which operates in both discrete time and frequency domains:

$$X(\omega) = \sum_{n=0}^{L-1} x(n) e^{\frac{-2\pi i k n}{N}} \tag{4.2}$$

In practical scenarios, it is also typical to ensure that the number of frequency points, N, in the transform (referred to as the frame size) is greater than the number of signal samples, L, within the window. When this condition is met, the DFT equation simplifies to:

$$X(\omega) = \sum_{n=0}^{N-1} x(n) e^{\frac{-2\pi i k n}{N}} \tag{4.3}$$

This form of the DFT no longer involves the window length, but solely the frame size. In

order to prevent using signal data beyond the intended window length, a technique called 'zero-padding' is often applied. Meaning, for all sample points where $n \geq L$, the signal value $x(n)$ is set to zero. By using this technique, an analysis frame that goes beyond the window's limits is created without adding any further information. Thus, the zero-padding technique allows that the frequency sampling resolution, representing the smallest detectable frequency in the frequency response, which depicts a system's response to different input signal frequencies, and the spectrum's representation via estimation, does not need to be overly small when analysing a narrow window of the signal [43]. The frequency sampling resolution is achieved by dividing the frequency range by the number of sampling points.

Still, DFTs are not typically used directly for the creation of spectrograms, which are time-frequency representations of a signal. This is due to their limitations in capturing both time and frequency information simultaneously, as it can be observed in eq. 4.3. Since audio data is the temporal development of frequency components, because audio data is dynamic, it is important to understand how the various frequency components change over time.

The Short-Time Fourier Transform (STFT)'s primary objective is to make it easier to go from a single, time-invariant representation to a series of representations that change over time. The understanding of dynamic changes in frequency components over a range of time intervals is made possible by this transition. At its core, STFTs can be understood as sequences of Fourier transforms applied to signals that have been windowed, providing the time-localised frequency information for situations in which frequency components of a signal exhibit temporal variations [64]. This transition is achieved by avoiding the application of the Fourier Transform across the entire signal duration. Instead, the signal is divided into smaller sections or frames, and for each of these frames, the Discrete Fourier Transform (DFT) is applied, figure 4.3.

The process begins with an audio signal, focusing on the first frame. At this point, only the samples within the frame undergo the DFT procedure, resulting in the magnitude spectrum, which represents the magnitudes of various frequency components. Subsequently, the DFT is again utilized for the next frame during the transition. This operation is repeated for all frames, covering the entire duration of the signal's segmentation. To derive segments from a signal, windowing can be employed along with the utilization of the zero-padding technique previously mentioned.

Earlier, when explaining the concept of STFT, it was mentioned that the signal undergoes division into smaller frames, with the DFT applied to each frame. However, this description is an oversimplification of the actual process. In reality, each frame is subjected to multiplication by the windowing function, and as the transition occurs from one frame to the next, there is partial overlap between them. This overlapping region is tied to another parameter, known as the hop size (H).

While the DFT calculation of $X(\omega_k)$ only takes into account specific frequencies (ω_k), the STFT of a signal $x(n)$, is mathematically represented as $X(m, \omega_k)$, taking into account both frequency

(ω_k) and 'time' (m):

$$X(m,\omega_k) = \sum_{n=0}^{N-1} x(n) \times w(n-mH) \times e^{-i2\pi n\frac{k}{N}} \tag{4.4}$$

where, $x(n)$ is the original signal at sample index n, $w(n-mH)$ is the window function applied to the signal for the frame m, H is the hop size, which indicates the shift between consecutive frames, N is the frame size, representing the number of samples in each frame, $e^{-i2\pi n\frac{k}{N}}$ represents the complex exponential term associated with the frequency and sample index, and $X(m,\omega_k)$ is the STFT of the signal at frame index m and frequency ω_k.

4.3.2 Outputs

While both DFT and STFT use mathematical transformations, their outcomes provide different insights. With DFT, a spectral vector - array of values representing the amplitude and phase of different frequency components within the analysed signal - is extracted. This means each Fourier coefficient in this 1-D array/ vector corresponds to a decomposed frequency component from the original signal (fig. 4.4). However, this lacks a temporal aspect, as it is averaged across the signal's entire duration.

STFTs, on the other hand, present a substantial difference. In this case, a frequency-bin and frame-inclusive spectral matrix (2-D array) arises. Within each frame, a Fourier coefficient is obtained for every frequency bin. Consequently, while information regarding the frequency details remains present, temporal perspectives are also obtained from distinct frames, functioning as proxies of time (fig. 4.5).

To determine the number of frequency bins within an STFT, the frame size is simply divided by two and add one.

Frequency bins of a DFT correspond to the complete samples in the signal. Due to STFT's consideration of a frame-sized sample portion, the number of frequency bins is expected to align with the frame size. However, due to the symmetrical structure of the DFT (fig. 4.4) [80], information from the first half suffices. The division by two and addition of one in the calculation of frequency bins for an STFT arises from the symmetric nature of the DFT and the redundancy in frequency information. The first half of the DFT spectrum contains meaningful information, while the second half is a mirror image. To avoid redundancy, only the first half is considered, and adding one accounts for the zero-frequency bin.

$$\#\,frequency\,bins = \frac{framesize}{2} + 1 \tag{4.5}$$

The number of frames, is calculated by subtracting the frame size from the total signal sample count and then dividing the result by the hop size, adding one at the end.

$$\#\,frames = \frac{samples - framesize}{hopsize} + 1 \tag{4.6}$$

4.3.3 STFT Parameters

Exploring the Short-Time Fourier Transform (STFT) and comprehending its parameters becomes crucial, as they significantly shape the outcomes. STFT's output varies according to provided parameters, underscoring the need to understand their functions. The STFT function has a number of potential parameters, according to the documentation provided by the *librosa* (Python package for music and audio analysis) library. There are a few parameters which stand out for their importance: the input signal, frame size, hop size, and windowing function are among them.

The frame size is a pivotal element in segmenting the original signal into discrete parts. The selection of different values for this parameter creates a trade-off between time and frequency information, fig. 4.6. Enlarging the frame size augments frequency resolution while reducing time resolution. This trade-off arises from the interaction between sample count, frequency, and time resolutions. A larger frame size yields more frequency bins, heightening frequency resolution. Yet, a larger time window due to increased samples leads to decreased time resolution. Conversely, a smaller frame size lowers frequency resolution but heightens time resolution, focusing on shorter time intervals with fewer samples. This trade-off lacks a definitive solution, relying on heuristics. Balancing resolutions is often pursued, with optimal choices dependent on the specific application. Some scenarios prioritize higher frequency resolution, while others prioritize precise time resolution.

The Hop size determines sample shifts when transitioning to new frames, fig. 4.3. Hop size values can be expressed as a fraction of the frame size, offering both absolute and relative possibilities. For example, the Hop size values can be expressed simply as 512, which is a fairly common value, or can be expressed instead as, for example, $\frac{1}{2}K$, being K the frame size.

Given that the STFT is a function of the signal's own characteristics as well as its interaction with the selected windowing function (eq. 4.4), the windowing function is also crucial. The original signal will be modulated by different windowing functions in distinct manners, which affects the STFT outputs.

4.3.4 Spectrograms

Since the 1980s, time-frequency representations of audio signals known as spectrograms have been used as input for neural network models [20]. In other words, spectrograms allow for the observation of sound. As stated in the subsections above, STFTs are matrices with each element within the matrix corresponding to a value that describes either the amplitude and phase of a specific frequency component of the signal (Fourier coefficient) or a complex number

representing the signal's characteristics at that point. These values collectively form the matrix representation of the signal's frequency content over time.
The time-frequency spectrogram Y_w is then defined as the squared magnitude of the STFT, as indicated by the following equation:

$$Y_w(m,\omega) = |X(m,\omega)|^2 \tag{4.7}$$

The equation 4.7 represents the calculation of the squared magnitude of the Short-Time Fourier Transform (STFT) coefficient for a specific frame m and frequency bin ω. In this equation, $X(m,\omega)$ represents the complex STFT coefficient at frame m and frequency bin ω, and $|X(m,\omega|^2$ is the squared magnitude of that coefficient. This squared magnitude gives information about the intensity or energy of the frequency component represented by that coefficient in the spectrogram.
Even though real values have replaced the complex ones, through the application of this equation, the resultant matrix, corresponding to the spectrogram values, still has the same form as the original STFT. Consequently, the transformation resulting from this equation grants the ability to visualise data using a heatmap, a visual map or graphic that employs colours to show and represent data values (fig. 4.7).
Focusing on the spectrogram itself, discrete time points are shown along the x-axis, and small discontinuities between frames, or temporal bins, are indicated by thin vertical bars. Along the y-axis numerous frequency bins are displayed, with its frequencies in Hertz. This visualisation granted by the spectrogram allows the observation of how various frequency components change throughout the course of the signal's frames, in other words, how they change over time.
There are different types of spectrograms for different applications [20]. The basic spectrogram, looked at so far, contains a linear nature which originates from two essential properties of the Fourier Transform: superposition and homogeneity. Superposition indicates that transforming a sum of functions equals the sum of their individual transforms. Homogeneity ensures that scaling the input signal proportionally scales its transform. These characteristics create a linear connection between the original signal and its Fourier Transform, resulting in the linear nature of the conventional spectrogram.
However, this linear representation poses a challenge regarding human frequency perception. Humans have finer resolution at lower frequencies than higher ones, more specifically to sound frequencies between 20 Hz and 20,000 Hz, or approximately 10 octaves [107]. Human frequency perception follows a logarithmic scale [92], unlike the linear scale used in traditional spectrograms. This discrepancy can limit the spectrogram's accuracy in capturing how humans perceive sound.
Consequently, basic spectrograms fall short in capturing perceptually (to humans) relevant frequency representation. This is where mel spectrograms, another type of spectrogram, come into play. The term "mel" is derived from "melody," closely tied to the notion of pitch, which is

a central element in melodies along with rhythm, and corresponds to a complete distinct frequency scale. The introduction of the mel frequency scale aimed to establish a measurement of pitch where equal differences in Mel-scale pitch equate to equal differences in perceived pitch, irrespective of the Hertz frequency [135].
Alongside Stevens et al.'s original Mel scale, multiple attempts have been made to refine it [134, 71]. As a result, there isn't a singular "correct" formula for the Mel scale, leading to the coexistence of various formulae in literature [147]. Nonetheless, one widely recognized frequency-to-Mel scale conversion is the method presented in O'Shaughnessy's book [105], as is represent below:

$$m = 2595 log_{10}(1 + \frac{f}{700}) \tag{4.8}$$

where m stands for mel and f for frequency.
Once the conversion of frequency to the Mel scale is established, Mel filter banks come into play. These banks divide the frequency spectrum into a series of overlapping triangular filters, each highlighting a distinct range of frequencies [72] (fig. 4.8). Put differently, the values within the mel frequency range do not represent equal frequency increments when viewed from a Hz scale. Instead, they correspond to equally spaced steps aligned with human auditory perception.
Examining fig. 4.8, let's take into consideration the instance of the point at 1526 mels, which aligns with 2000Hz. The gap between centre points of mel bands maintains a consistent measure, in contrast to frequency representation in Hz. This discrepancy is the key aspect – expressing frequency in Hz requires spacing out higher frequencies to achieve uniform perceptual differences in frequency distances.
Focusing solely on constructing a single triangular filter for a mel band, let's take the point at 1526 mels, present in fig. 4.8, as the centre of the mel band. To determine its lower and upper boundaries, the centres of the previous and subsequent mel bands are examined. Following the trajectory of the green line, it starts below 1263 mels (centre of the previous mel band) and ends below 1789 mels (centre of the subsequent mel band).
For frequencies below the lower boundary and above the upper boundary of a mel band, the weight is set to zero. This essentially mutes the signal below and above these frequency ranges. The next step is to connect the lower and upper boundaries with the centre frequency, where the weight is set to one. This process results in the creation of a triangular filter.
Repeating this procedure for subsequent mel bands, leads to the complete mel filter bank, consisting of triangular filters. These filters serve a crucial purpose – they can be applied to a regular spectrogram, allowing the isolation of different frequencies and their conversion from normal frequencies in Hz to frequencies expressed in mels.
Representation of a Mel filter bank extends beyond just visual means, it can also be captured as a two-dimensional array. This array's shape is determined by the number of bands (specific ranges of frequencies used in the Mel scale) in the Mel filter bank and the frame size divided by

two plus one, which was defined before as the number of frequency bins.
In order to go from the basic spectrogram to the mel spectrogram it is necessary to apply the mel filter banks to the spectrogram. With the shape of the Mel filter bank matrix denoted as $M = (\#bands, \frac{framesize}{2} + 1)$, and the matrix shape of the basic spectrogram as $Y = (\frac{framesize}{2} + 1, \#frames)$, the number of rows of the matrix of the basic spectrogram is equal to the number of columns of the mel filter bank matrix. Thus, matrix multiplication can be done.
The result of the multiplication of these matrices results in the shape of the mel spectrogram matrix. This means, the mel spectrogram itself is a matrix and its shape is given by:

$$Melspectrogram = MY \tag{4.9}$$

$$MY = (\#bands, \#frames) \tag{4.10}$$

Therefore, the number of the matrix rows corresponds to the number of bands chosen, and the number of columns is equal to the number of frames of the basic spectrogram (fig. 4.7).
In essence, employing Mel filter banks on a spectrogram (MY) facilitates the conversion of frequency units from Hertz to mel bands. While further insight into the computation of Mel filter banks is available in [24, 72], a comprehensive explanation is beyond the scope of this thesis and thus not provided here.
Visually, when examining Figure 4.9, mel spectrograms bear resemblance to basic spectrograms (fig. 4.7). The Mel spectrogram outcome retains the structure of a heat map, mapping 'time' along the x-axis and 'frequency' along the y-axis. However, in the Mel spectrogram, distinct frequency bins correspond to mel bands, that better represents how humans perceive sound.

4.4 Deep Learning in Bioacoustics

Bioacoustics has historically benefited from computational analysis methods such as signal processing, and machine learning techniques [143, 141]. The recent rise of deep learning, which has roots in artificial intelligence developments made for text or image processing, has culminated in a moment of transition for several computational fields [76, 46]. This innovative influence is now spreading to the area of computational bioacoustics, where deep learning is demonstrating its ability to function as a catalyst for solving previously challenging issues, such as species identification for example [132]. The enormous increase in data accessibility in the twenty-first century, driven by widely accessible digital recordings, effective data storage, and sharing platforms, enables and compels this shift [118, 123].
The difficulties presented by this abundance of audio data, such as the scarcity of human resources for manual analysis, highlight the need for techniques that may automate important workflow steps, such as machine learning. The journey of deep learning in bioacoustics is rela-

tively recent and has seen remarkable innovations. While deep learning has been used for audio tasks for over a decade, its application in wildlife bioacoustics is recent and remains in its early stages [136]. In 2017, Hershey *et al.* made a significant contribution to audio recognition with convolutional neural networks. This paper introduces the AudioSet dataset, containing 1 million 10 second excerpts labeled with a vocabulary of acoustic events, and the widely used VGG architecture, which corresponds to a convolutional neural network that made improvements over the *AlexNet*, responsible for winning the ImageNet challenge (mentioned in section 2.2) [52].

Deep learning's adaptability makes it appropriate for a variety of applications, including classification, regression, signal augmentation, and even data creation [73, 23, 128, 47]. Despite its versatility, classification, which involves assigning labels to data objects from prepared lists, remains an important aspect of deep learning. Numerous advancements in deep learning have centred around tasks related to classification, encompassing a wide range of applications in the field of computational bioacoustics.

Species' data from numerous taxa have had deep learning based vocalisation analysis applied to them. This entails birds, as the most extensively studied group [137, 62], marine mammals like cetaceans [40, 130], terrestrial mammals including bats [41], primates [29], and elephants [16], anurans [75], insects [131], and fish [153]. The importance of some taxa is a result of a number of factors, including their conservation significance, behavioural research, and the complex nature of their vocalisations.

There are some previous studies that have worked on implementing automatic detection of beaked whales. In a study by Rankin *et al.*, [119], a hierarchical random forest event classifier called BANTER (Bio-Acoustic eveNT classifiER) was applied to large acoustic datasets of beaked whale detections resulting from surveys conducted in regions of the Atlantic and Pacific. A random forest is a machine learning technique that builds numerous decision trees and then combines their predictions. A hierarchical random forest expands on this by including a hierarchical component, generating predictions progressively from broad to narrow categories. Ziegenhorn *et al.*, in [161], is another study that focused on classifying odontocete echolocation clicks in the Hawaiian Islands through the application of a machine learning toolkit using unsupervised clustering methods and combining those with human-mediated analyses.

To this day, there is no general consensus on the standard procedure for applications of deep learning to bioacoustic problems. However, all researchers follow certain pipelines that are generally considered correct [136]. Any study involving deep learning in bioacoustics starts with a task definition where the specific classification task is decided, such as identifying species, individuals, call types, or detecting the presence of certain sounds. For this, a dataset is needed, so several things must be taken into consideration: data collection, data pre-processing, data labelling, and data splitting.

Taking into consideration past literature, the methodologies and processes that are going to be outlined will serve as a reference for applying deep learning techniques to address bioacoustic

challenges.

When collecting data in bioacoustics, recording equipment, like microphones, is used to record animal vocalisations in their habitats. For instance, Dufourq *et al.*, in [29], attached recorders to trees in tropical evergreen forests, placing them within the known home range of Hainan gibbons, at intermediate locations, and in areas further where solitary males were thought to occur.

Once the data has been collected, a pre-processing step is applied next where the raw signal is typically converted into a spectrogram, which will later serve as the model's input data. Typically, the spectrograms correspond to segments of the original raw audio that were fixed sized to around 1 second to 10 seconds. The spectrograms may be linear-frequency (representing the time on the x-axis, the frequency in Hz on a linear scale on the y-axis, and the power in decibels), illustrated in [160], mel-spectrograms (using the mel scale instead of frequency in Hz on the y-axis), illustrated in [102], or log-frequency spectrograms (the y-axis corresponds to the frequencies on a logarithmic scale rather than a linear one), illustrated in [154].

Once the raw audio signals have been transformed into a visual representation, such as spectrograms, data annotation can take place. The set of labels that need to be thought of could relate to call types, individuals, species, or something else. When the training dataset is small, usually data augmentation is applied to increase the sample size by adding artificial samples originated from the manipulation of existing ones, as described in sub-section 5.2.6. This pre-processing step not only increases the size of the dataset but also makes it more diverse, for example, by mixing noise (adding random noise to the existing samples) or time shifting (slightly changing the timing or position of the data samples), as demonstrated by Dufourq *et al.*, in [29].

After the dataset has been labelled, the researcher decides if, given the characteristics of its dataset, it needs to augment a certain class to try to balance the dataset more or not. This is followed by data splitting, where a percentage of data is allocated for training, validation, and testing. Data split always varies between studies but is commonly split around 60-80% for the training set and 10-20% for both the validation and testing sets. Zhong *et al.* illustrate this in [160], where in order to classify beluga whale acoustic signals using deep neural networks, the data was split into 49% for training, 21% for validation, and 30% for testing. Bergler *et al.* in [13] uses 4 different datasets, all of which were split in different percentages. Namely, the OAC dataset was split into 69.9% for training, 14.9% for validation, and 15.2% for testing. The AEOTD dataset was split into 80.2% for training, 9.9% for validation, and 9.9% for testing. The DLFD dataset was split into 74.8% for training, 12.9% for validation, and 12.3% for testing. And the SUM dataset was split into 75.5% for training, 12.4% for validation, and 12.1%. Thus, these examples show the variability in the percentage split of the datasets depending on their characteristics.

The architecture of the model is the next essential component to be aware of. CNN is by far the most common model used in deep learning for bioacoustic tasks. Usually, when deciding

on the model's architecture, researchers take one, or both, of the following approaches: create a CNN model architecture or build upon pre-trained model weights, which is described as *transfer learning*. Noumida *et al.*, in [102], exemplifies a case where both approaches were followed and later compared. When training a deep learning model, it is considered good practice, according to Goodfellow *et al.*, in [46], to use optimisers (usually Adam), dropout layers, apply hyperparameter tuning, and do an early stopping (a technique to prevent overfitting by monitoring the performance of the model on a separate validation dataset during the training process) if needed.

When the model has been trained and tested, typically deep learning models in bioacoustic problems use metrics such as accuracy, recall, specificity, precision, and the F1-score to evaluate the test set and determine the model's performance. The number of metrics used to evaluate the model's performance and the particular metrics used vary depending on the answer that the researcher is trying to find. Nonetheless, as shown in [102, 29, 160], these are frequently used when tackling deep learning for bioacoustic classification problems.

Thus, there is a lack of consistency amongst the literature; however, some high-level trends, as is done in computer vision tasks, are followed.

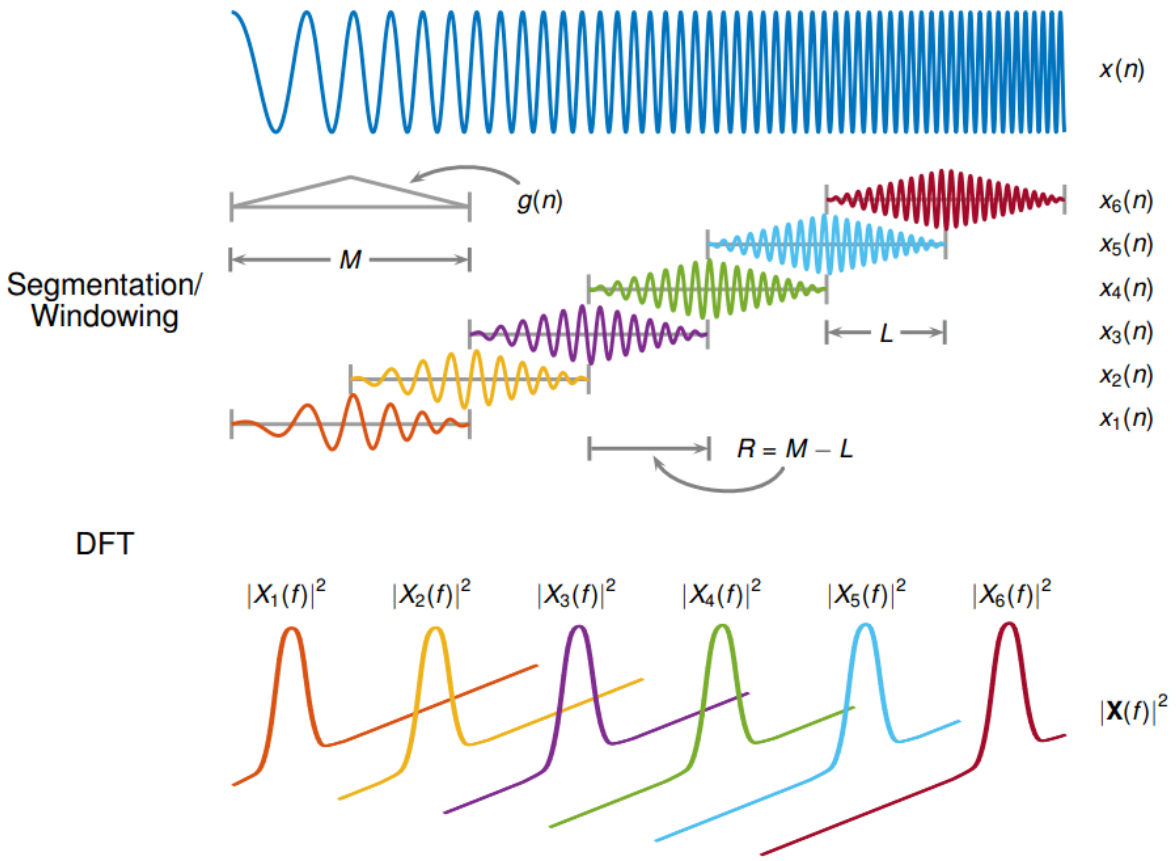


Figure 4.3: Representation of the Short-Time Fourier Transform (STFT) of a signal, determined by moving an analysis window $g(n)$ with a length of M through the signal. The discrete Fourier transform (DFT) is computed for the segment of data covered by the window, at each step. This process involves sliding the window over the original signal with a step size of R samples, resulting in an overlap of $L = M - R$ samples between adjacent segments. The DFT of each windowed segment is then accumulated into a complex-valued matrix, which contains both the magnitude and phase information for each point in the time-frequency domain. Image adapted from Matlab Short-Time Fourier Transform documentation.

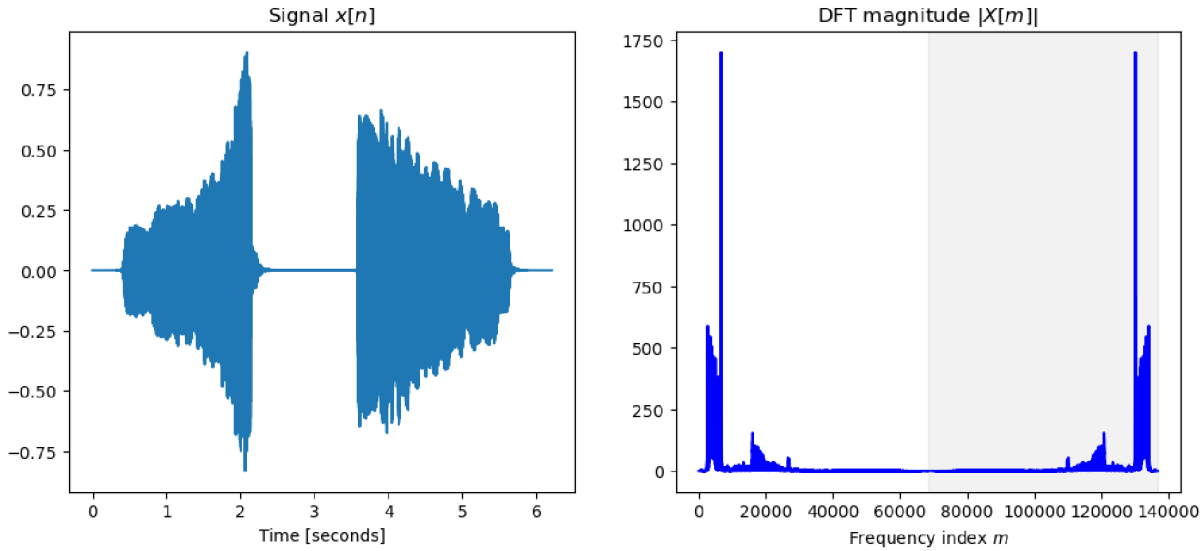


Figure 4.4: Sampled signal at a frequency sample of 22050 audio samples captured per second. The left segment of the illustration showcases the raw waveform shape of the signal denoted as $x[n]$. While this depiction captures the temporal evolution of the amplitude, it lacks the capacity to unveil any insights regarding the signal's frequency composition. Contrarily, the right segment presents the magnitude spectrum plot, elucidating the magnitudes of distinct frequencies inherent within the signal. The negative frequency range is shaded in gray, which portrays the symmetrical nature of DFTs. Usually this half can be discarded as it does not provide any new relevant information.

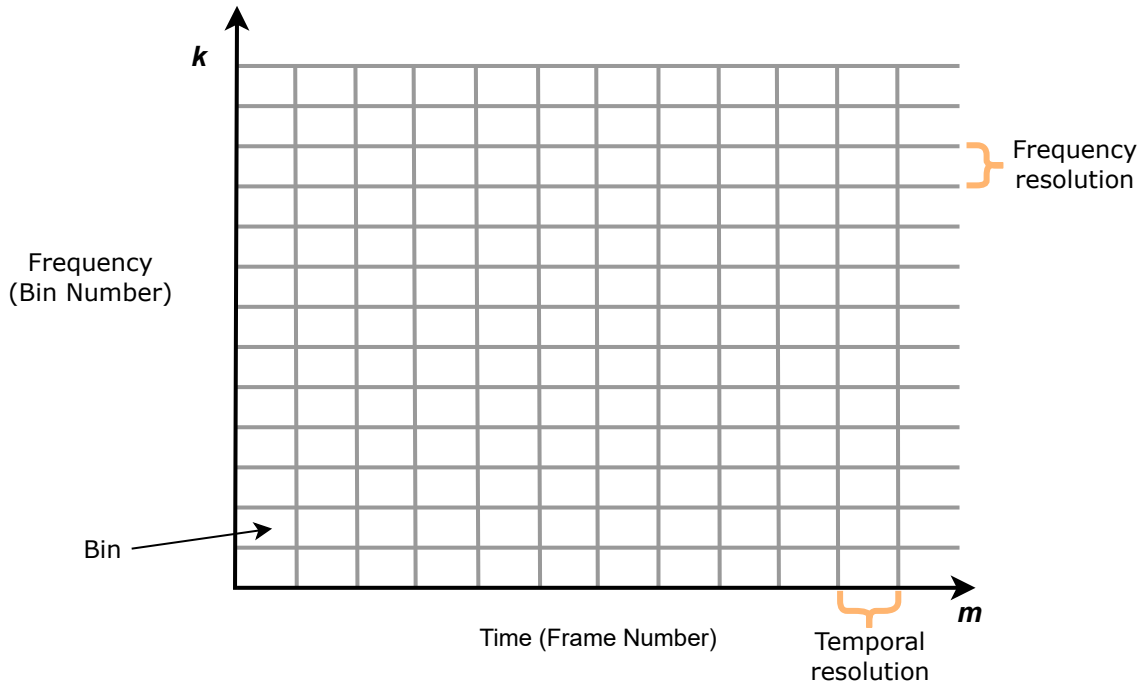


Figure 4.5: The provided figure serves to elucidate the intrinsic time-frequency characteristics of Short-Time Fourier Transforms (STFTs), culminating in a two-dimensional representation. This representation encompasses both frequency and temporal data. The window length is proportional to the resolution cell in time, indicated by the vertical lines. The width of the dominant frequency component of the window-transform is proportional to the resolution cell in frequency, indicated by the horizontal lines.

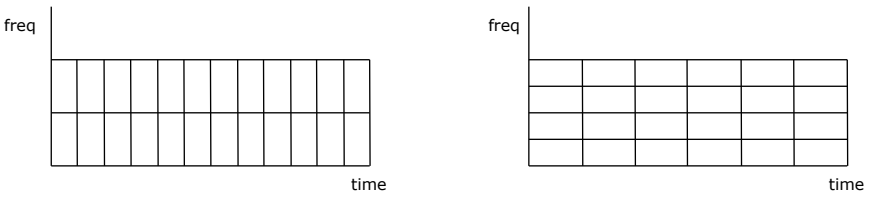


Figure 4.6: The following image serves to illustrate how a trade-off between frequency and time resolution is present within STFTs.

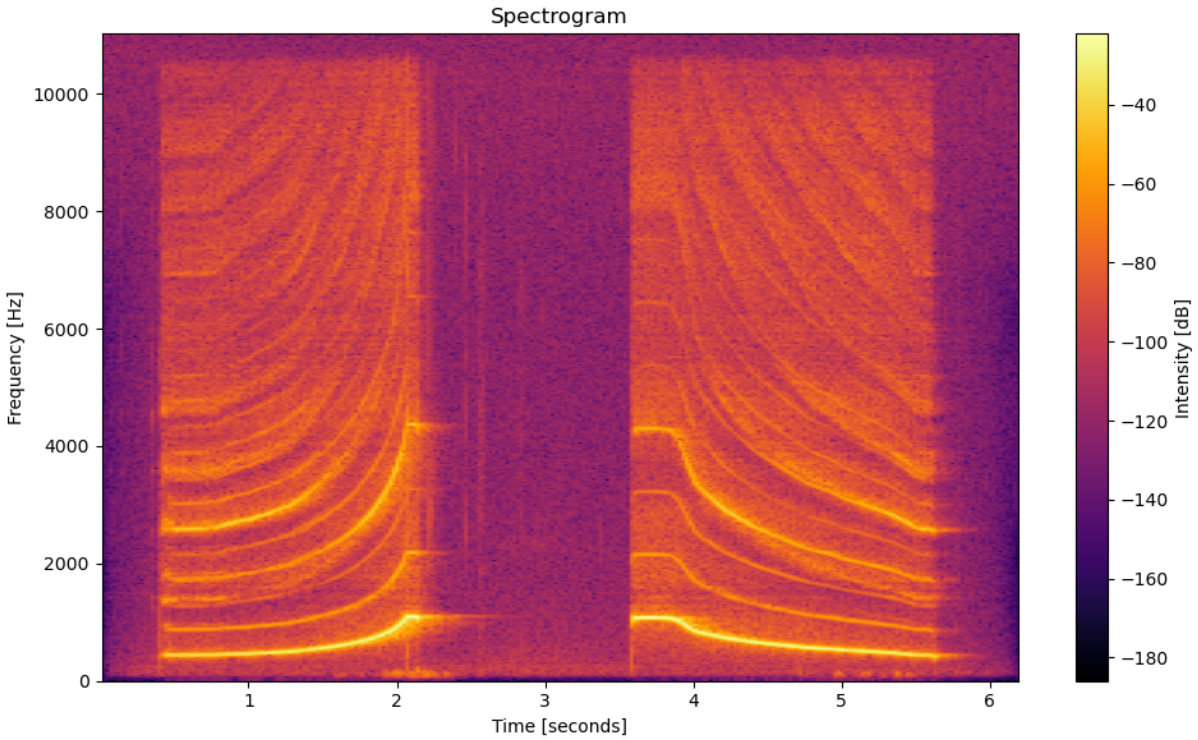


Figure 4.7: Illustration of a spectrogram, with the y-axis representing the frequencies on a Hertz scale, and the x-axis representing time in seconds. The bar on the right of the spectrogram gives the intensity of the signal in decibels, a logarithmic unit used to measure the intensity or level of sound or signal, through a colour code, where brighter colours represent higher intensities.

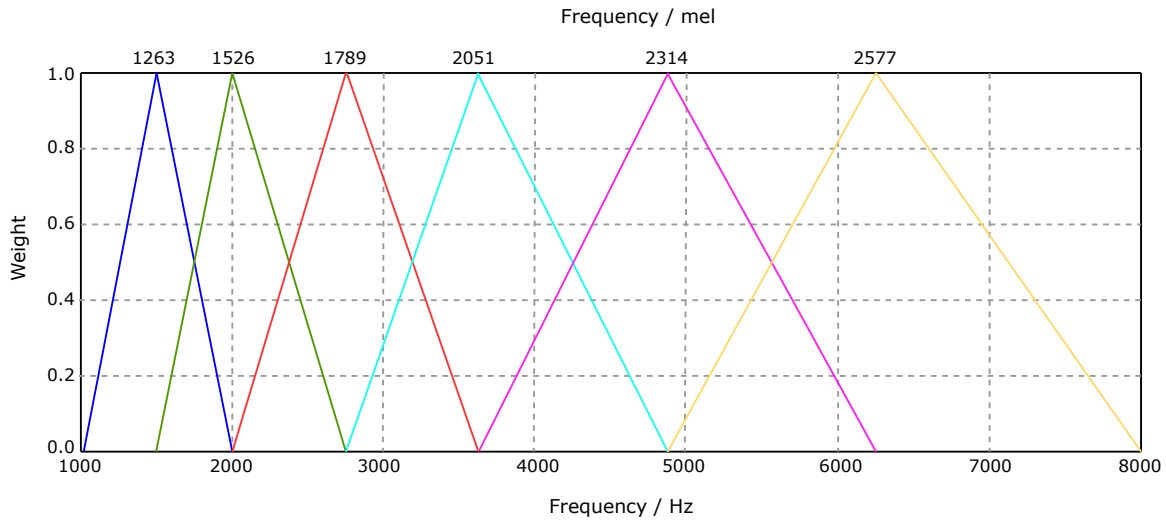


Figure 4.8: This illustration showcases six mel bands, accompanied by their corresponding frequency values in the mel scale. These values denote the central frequencies for the respective mel bands. The x-axis displays Frequency, with the lower part representing Hz and the upper part representing mels. Weight is represented on the y-axis, ranging from 0 to 1. These weights act as filters for the sound. A weight of one indicates that the entire signal remains unchanged, while decreasing weight values correspond to increased sound filtering.

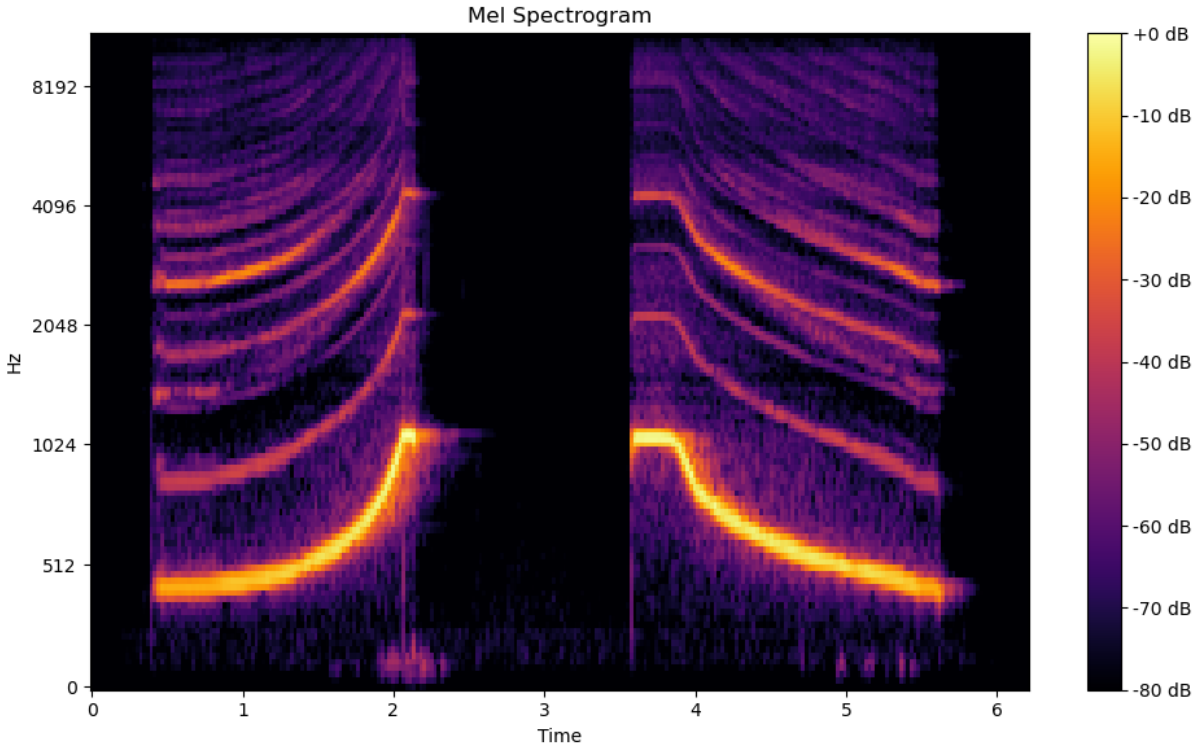


Figure 4.9: Illustration of a mel spectrogram, with the y-axis representing the frequencies on a Hertz scale, and the x-axis representing time in seconds. Even though the values on the y-axis are portrayed in Hz, these result from the conversion of mel back to Hz. The bar on the right of the spectrogram gives the intensity of the signal in decibels, a logarithmic unit used to measure the intensity or level of sound or signal, through a colour code, where brighter colours represent higher intensities.

Chapter 5

Methodology

5.1 Introduction

This chapter comprehensively details the approaches and decisions underpinning the modelling segment of this thesis. This comprises assessing the data, performing any necessary pre-processing, and controlling any concerns pertinent to data quality. Furthermore, this chapter delineates the creation and implementation of the classification model.

5.2 Data handling

5.2.1 Data Collection

Data collection is a crucial phase of the project since it precedes the actual application of the model, and every aspect of it determines the trustworthiness of the results. The data used in this thesis were obtained through the ACCURATE Project, wherein data gathered from DTAG recorders attached to various whale species were available. For this thesis, the specific focus of the data collection lies on the beaked whales, in particular Blainville's and Cuvier's beaked whales, given that they were one of the species with a vast amount of DTAG recordings available. DTAGs are animal borne tags with suction cups for attachment. This attachment procedure involves utilizing a specialised pole equipped with the DTAG at its tip, which, when a whale surfaces to breath, is adhered to the dorsal region of the whale. The duration of data recording by the tag was contingent upon its memory capacity and the rate at which it sampled audio data. All the audio captured in the DTAGs was in mono format, a common approach ([13, 137]). Data collection for Blainville's beaked whales (*Mesoplodon densirostris*) covered the period from 2003 to 2017. For Cuvier's beaked whales (*Ziphius cavirostris*) data was collected from 2003 to 2013. Blainville's beaked whales were tagged in distinct locations, including the Bahamas and El Hierro in the Canary Islands. Similarly, Cuvier's beaked whales were tagged in two separate

geographical regions, namely Liguria in Italy and Southern California.
In order to avoid an early release due to leakage between the suction cups and the whale, these DTAGs attachments included a pump powered by the pressure changes during the whales' dive cycle to maintain vacuum in the cups. An active release was also included consisting of a nickel-chromium wire which seals a valve in the air line to each suction cup. The wire corrodes rapidly in seawater and is controlled by a clock circuit in the DTAG. Meaning, the suction cups lose the vacuum fixating them after a programmable time to release the tag from the animal, which is then recovered to download the data [59]. Once detached from the animal, the DTAG would float at the surface of the water. To aid in retrieval, it emitted a very high frequency (VHF) radio beacon [59], enabling the location and recovery the tags.
The sampling rate at which acoustic data was recorded varied with deployment year. Older DTAGs used a sampling rate of 96kHz, while DTAGs from around 2004 to 2015 used a sampling rate of 192kHz. The most recent DTAG deployments, spanning from 2015 to 2017, had a 240kHz sampling rate. A total of 31 deployments were conducted on Blainville's beaked whales, generating 357 audio files for analysis. Similarly, 12 deployments were carried out on Cuvier's beaked whales, resulting in a total of 168 audio files available for analysis.

5.2.2 Data Assessment

The audio files extracted from DTAG recordings were stored in ".wav" format. However, as the recordings grew in size, particularly with more recent ones (dating from 2015 onwards) sometimes surpassing 100 Gigabytes, efficient management became imperative. To address this challenge, each animal's recordings underwent segmentation into multiple shorter files, aiming to reduce their size to a more manageable one, typically around 3 to 4 Gigabytes each. This division helped make the processing of files more efficient because handling such large files requires a lot of processing power and memory. These files were then systematically categorised within folders, the names of which adhered to a precise naming convention. The folder name commenced with a two-letter code reflecting the species' scientific nomenclature, followed by a two-digit number representing the year. These components were separated from a three-digit Julian date — a continuous count of days since the beginning of the year — by an underscore. Additionally, a letter was consistently appended to the folder name, at the end, to identify individual animals, in case there were tagged multiple animals on a given day. This methodical naming scheme was done so that an efficient identification of the species, year, month, day, and individual associated with each recording was possible. The audio files, present in these folders, were labelled beginning with the species' two-letter code, followed by the three-digit Julian date, the individual's identifying letter, and a sequence of two or three numbers that indicated the chronological order of the segmented ".wav" files. As an example, whale *md15_156a* generated files *md156a001.wav*, *md156a002.wav*, and so on until it reaches the final ".wav" file, in this case, *md156a019.wav*.

5.2.3 Data quality

Data integrity was a priority in this project, prompting specific measures to manage corrupted audio files and missing annotations. When a tag contained corrupted ".wav" files, all subsequent files were excluded. However, missing annotations for a certain dive did not warrant exclusion, because the model primarily relied on audio files and their corresponding timestamps, that is, missed annotations meant the model would not train on those specific instances.
At the start of the project, the dataset comprised three DTAGs for Blainville's beaked whales. Following the inclusion of additional data facilitated by the ACCURATE project, the total number of DTAGs increased to 32 for Blainville's beaked whales and 16 for Cuvier's beaked whales. This brought the initial count to 48 DTAGs after integrating the supplementary data. However, during data processing, some DTAGs encountered issues such as corrupted ".wav" files or absence of any annotation. As a result, the final number of DTAGs used decreased to 31 for Blainville's beaked whales and 12 for Cuvier's beaked whales. This totalled 43 DTAGs after factoring in the incorporation of additional data and the exclusion of non-optimal DTAGs.

5.2.4 Data Annotation

The primary objective was to develop a click detector by analysing the acoustic characteristics present within the recorded data. This involved distinguish clicks from other ambient sounds through detailed analysis. In order to accomplish this, custom Matlab scripts (".m" type files), previously created by Mark Johnson, were used, applying both depth data and audio recordings collected by the tag, specifically focusing on deep dives, defined as dives greater than 200 metres. The aim of this was to determine instances when the whale produced distinctive clicks during these deep dives.
The process of click annotation involved a two-step approach, employing functions developed by Mark Johnson [freely available online ([60])]. These functions were used as a preliminary tool to help navigate through the audio files and manually annotating them, speeding up the annotation process.
Initially, the "d3findallclicks" function was executed to identify probable clicks emitted by the tagged whale. However, it is crucial to note a significant difference between this tool and the developed model in this study. The tool proposed by Johnson was created to identify clicks from any tagged whale, discarding surrounding whale clicks even if they are from conspecifics (members of the same species), while the model developed in this thesis was created to detect beaked whales, and employed a deep neural network for this purpose. The "d3findallclicks" function involved specifying essential parameters such as the tag's name, the folder's path directory of the tag's data, and the specific time/cues period set for the function to examine in search of click events. Thereafter, the results produced by the "d3findallclicks" function were carefully reviewed using the "d3findmissedclicks" function. This second stage involved distinct parame-

ters, like the tag's name, the variable containing the data obtained from the "d3findallclicks" function, the chosen frequency range (both minimum and maximum frequencies in Hz) used for filtering the signal, and specific criteria such as the shortest gap between Inter-click Intervals (ICIs) to consider, along with the minimum fractional change in ICI deemed as a significant gap, which in other words refers to the predetermined threshold or standard established to determine the minimum acceptable difference in duration between successive clicks.

The "d3findmissedclicks" function presents an interface window, displaying spectrograms corresponding to brief time windows within the entire tag data, as shown in the Appendix section 7.2. Here, the function marks potential clicks with a small red circle atop distinct features within the spectrogram that it identifies as clicks. These spectrogram windows slide across 5-second intervals, systematically examining the entirety of the tag's audio data. Consequently, the primary objective of the "d3findmissedclicks" function is to manually rectify any misclassifications made by the "d3findallclicks" function. In essence, this function is utilised to add missed clicks or remove incorrect classifications, refining the accuracy of click identification within the dataset.

The clicks were manually verified and adjusted, using *Sonic Visualiser*, by visualising spectrograms generated by the "d3findmissedclicks" function, and simultaneously listening to the audio files. This combined approach ensured precise annotations were made by cross-referencing the visual and auditory data, although with annotations done manually bias and human error can always occur.

This process was initially carried out for multiple DTAGs. The average time taken for annotating was approximately two and a half hours per hour of audio, subject to variations based on the noise levels during each dive, making this one of the most time-consuming components of the whole project. However, the primary focus was directed towards extracting features from audio recorded during deep dives, as whales did not produce clicks at shallower depths, leading to a more concentrated effort on these specific dive segments. Following the annotations, the generated data were stored as ".mat" files, comprising timestamp information for each whale-produced click.

Later, the ACCURATE project provided an additional set of annotated DTAGs, expanding the dataset to a total of 48 tags — 32 for Blainville's and 16 for Cuvier's beaked whales. These supplementary DTAGs underwent a similar click annotation process, although conducted by people connected to the ACCURATE project, employing a comparable methodology to ensure consistency.

After incorporating the additional annotated DTAGs from ACCURATE, this dataset significantly expanded from 51.5 to about 311.8 hours for Blainville's beaked whales and approximately 103 hours for Cuvier's beaked whales, totalling 414.8 hours of audio recordings for both species.

5.2.5 Data analysis

The initial phase of data preparation involved manual annotations of each recorded ".wav" file, ensuring precise alignment with their respective DTAGs. Subsequently, both the audio files and annotations were imported into Python for dataset construction, essential for the subsequent machine learning model development.
In order to ensure that the input data of the CNN was uniform in size and format, the recordings were segmented, dividing each into fixed-length segments of 2 seconds. Two-second windows were chosen because they are long enough to capture entire click events, reducing the risk of clicks being split between segments, which in turn reduces the risk of misclassification.
Each segment was labelled based on cross-referencing its time intervals with the annotated whale click times. Segments were classified as:

- "Blainville's beaked whale click", if there were any Blainville's beaked whale clicks present in the segment;
- "Cuvier's beaked whale click", if there were any Cuvier's beaked whale clicks present in the segment;
- "No-click", otherwise.

There were no instances of aliasing detected in this process. Aliasing, which occurs when frequencies outside the intended range disrupt the recording, was not observed. Therefore, there was no need for any anti-aliasing actions to be taken.
In this thesis, two different experiments were conducted: one to evaluate the model's classification capabilities and another to compare the model with the original Matlab tool. Depending on the experiment, the audio was processed differently.
In order to evaluate the model's classification capabilities, the audio segments were transformed into mel-spectrograms ([13]), serving as input images for a 2-D CNN. Utilising a Hann analysis window size of 3072 samples (32 ms) and a hop size of 256 samples (5.312 ms, 91.67% overlap), the transformation generated 128 mel frequency bins spanning frequencies from 1000 to 48000 Hz. The computation of these mel-spectrograms, conducted through the Librosa library, resulted in images sized at 128×751 pixels. A Hann analysis window (the default window used in Librosa), employed in this process, functions by tapering the edges of signal segments, ensuring a smoother frequency analysis. These images comprised 128 frequency bands, meaning each of the 128 distinct vertical bins represents a specific frequency range within the audio signal, and 751 time steps, where each step represents the sequential time frames or segments of the audio signal, capturing the evolution of the sound signal over time.
In order to compare a model with the original Matlab tool, the audio files were segmented by dividing each into fixed-length segments of 0.08 seconds, instead of the 2-second segments earlier mentioned. This specific window length was chosen to align with the species' inter-click

interval, ensuring on average a single click per window. This choice enables a comparison with the original Matlab tool, as is discussed in Section 5.4, and facilitates the estimation of densities through the implementation of methods such as the one used by Marques [89]. Employing a Hann analysis window size of 512 samples (equivalent to approximately 5.33 ms) and a hop size of 16 samples (equivalent to 31.98 ms, 96.88% overlap), the transformation produced 50 mel frequency bins encompassing frequencies from 1000 to 48000 Hz. Using the Librosa library for computation, these mel-spectrograms resulted in images sized at 50×481 pixels.

5.2.6 Data Augmentation

Data augmentation is the process of creating new samples by applying adjustments to already existing ones, hence expanding sample sizes. Translations and rotations are two common geometric operations used in this method. The objective of its common application is to improve classifier performance, particularly in situations when the amount of the training dataset is limited ([53]).

In order to correct the class imbalance within the dataset, data augmentation was implemented. To rectify the raw proportions across different classes and balance the dataset, two methods were adopted. First, samples for the over-represented classes were reduced to provide a more equal distribution of representation across all classes. Second, two different approaches were used to apply augmentation to the underrepresented classes.

The augmentation process encompassed two types: noise augmentation and addition of random noise samples to the existing click data, in order to augment the clicks of a certain class if needed. In noise augmentation, column order randomization, of each spectrogram 2-second window, was utilized to introduce variability within the dataset. Moreover, random noise samples were incorporated specifically into the click data of underrepresented classes to expand and diversify their samples.

Importantly, the loading of click samples incorporated a shifting mechanism, utilising a uniform distribution. This ensured that each instance of loading a click sample involved a shift, contributing to a greater diversity in the augmented dataset. Overall, these augmentation strategies were deployed to balance the dataset's class representation and enrich the variability of the training data for improved model performance.

5.3 Performance Evaluation

This section is focused on how the performance of the classification model is evaluated during the training and the testing.

5.3.1 Categorical Cross-entropy

One widely used loss function for multiclass classification scenarios is the *categorical cross-entropy*. This particular loss function measures the differences between predicted probability distributions and the actual categorical distributions, typically employed alongside the *softmax* activation function in the output layer. As mentioned previously in sub-section 2.3.3, the *softmax* activation function transforms a vector of real numbers into a probability distribution, which guarantees that values fall between 0 and 1 and that their sum equals 1 [39]. By representing the input vector, output vector, and predicted network output as, $\mathbf{x} = [x_1 \dots x_i \dots x_n]$, $\mathbf{y} = [y_1 \dots y_k \dots y_o]$, and $\hat{\mathbf{y}} = [\hat{y}_1 \dots \hat{y}_k \dots \hat{y}_o]$ respectively, the mathematical expression for this loss function is formulated as follows:

$$-\sum_{k=1}^{N} y_k \ln(\hat{y}_k) \tag{5.1}$$

When using this loss function the objective is to minimise the categorical cross-entropy in such a way to make the network predictions as similar to the labels as possible. In this case, the target labels will be represented as a *one-hot encoded* vector and the network predictions will also be in a vector of the same length. *One-hot encoding* is a method for representing categorical variables as binary vectors. Each unique category is assigned a binary vector where only one element is set to 1, indicating the presence of that category. In the context of this thesis, a classification problem involves three distinct class values, leading to the utilisation of one-hot encoded vectors with a length of 3, where '0' is represented as [1 0 0], '1' as [0 1 0], and '2' as [0 0 1].
Consider a scenario where $y_k = [1\ 0\ 0]$ for a specific sample. Assuming that a neural network N_1 predicts $\hat{y}_k = [0.90\ 0.05\ 0.05]$, in this case the categorical cross-entropy for sample k would be $-\ln(0.90) \approx 0.11$. Now, assuming another network N_2 predicted $\hat{y}_k = [0.30\ 0.60\ 0.10]$. In this case, the categorical cross-entropy for this sample would be $-\ln(0.30) \approx 1.20$. This means, N_1 is favoured since $0.11 < 1.20$, indicating that it provides a more accurate prediction for this specific sample according to the categorical cross-entropy criterion.

5.3.2 Confusion Matrix

A *confusion matrix* is a tabular representation that shows the counts of *true positive* (TP), *true negative* (TN), *false positive* (FP), and *false negative* (FN) predictions. Its purpose is to summarise the performance of a classification model, where each row represents the actual class and each column represents the predicted class (figure 5.1). *True positives* are examples of when the model predicts the positive class correctly. *True negatives* are examples of when the model predicts the negative class correctly. *False positives* are examples of when the model makes incorrect positive predictions (Type I error). *False negatives* are examples of when the model makes incorrect negative predictions (Type II error).

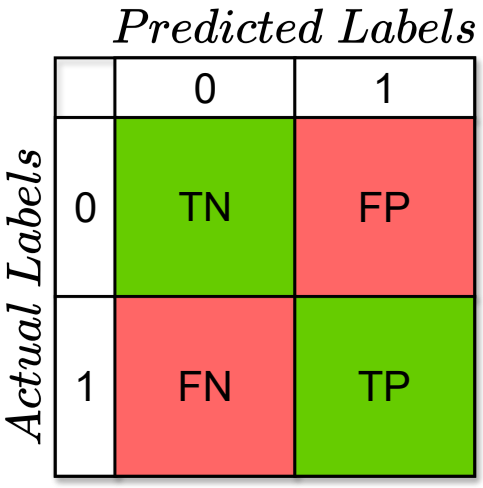


Figure 5.1: Representation of a confusion matrix.

Confusion matrices are often a preferred metric over accuracy, particularly in scenarios with class imbalance [11]. As accuracy represents the percentage of correct predictions made by a model by comparing predicted values to the actual values (eq. 5.2), it can be misleading especially in data sets with imbalanced classes.

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \tag{5.2}$$

This is due to the fact that accuracy does not account for the distribution of the data [111]. Moreover, the confusion matrix enables the calculation of metrics that offer deeper understanding of a model's performance. One such metric is the *recall* (Rec), also known as *sensitivity* or *true positive rate*, which shows the ability of the model to correctly classify instances that belong to the positive class.

$$Rec = \frac{TP}{TP+FN} \tag{5.3}$$

Specificity (Spec), another important metric, measures the ability of the model to correctly identify negative cases out of all actual negative cases in the dataset:

$$Spec = \frac{TN}{TN+FP} \tag{5.4}$$

The *false positive rate* (FPR), which is derived from specificity, quantifies the proportion of negative cases that are incorrectly classified as positive out of all actual negative cases in the dataset:

$$FPR = 1 - SP = \frac{FP}{FP + TN} \tag{5.5}$$

Precision represents the proportion of the positive predictions that were correctly classified:

$$P = \frac{TP}{TP + FP} \tag{5.6}$$

Lastly, the *f1-score* represents the harmonic mean (a type of mean that balances the influence of each individual value by considering their reciprocals) of the precision and recall, combining both metrics into a single value. This formula is defined by:

$$F1 - score = \frac{2}{\frac{1}{P} + \frac{1}{Rec}} = 2 \times \frac{P \times Rec}{P + Rec} \tag{5.7}$$

5.4 Model development and implementation process

The development and implementation of models for whale classification relied on spectrograms extracted from tagged whale audio recordings. Manual annotations of whale-produced clicks were categorised as one of the three following classes: "Blainville's beaked whale clicks", "Cuvier's beaked whale clicks", and "No-click". This labelled dataset was subsequently used to train CNNs for spectrogram classification.

During the initial exploratory phase, five distinct architectural variations (Models 1—5, Fig.5.2) were selected, each built with different configurations. From these five architectures, the one identified as model 1 was based on the one described in the literature [14]. The architecture identified in figure 5.2 as model 2 was based on the architecture described in the literature [29]. The remaining architectures resulted from different combinations of model 1 and model 2 architectures. Key hyperparameters such as learning rate, batch size, epochs, buffer size, and window size remained unchanged between the different architectures. Modifications were exclusively targeted at other components of the model, such as the number of layers, activation functions used, and size of the filters applied, ensuring that changes in the model performance were due to the architectural changes and not to hyperparameters. After randomly selecting one subset for training and one subset for validation, the testing subset was made up of an independent folder containing multiple .wav files. The exact same training, validation, and testing subsets were used for all architectures, and the architecture with the greatest performance metrics was chosen. This architecture ended up being the one identified as model 3, shown in figure 5.2.

The training process involved using the selected architecture on labelled spectrograms for 20 epochs. This number was determined through the visualisation of the evolution of performance metrics during training and validation to ensure the model converges well without excessively fitting to the training data, in other words, without overfitting. The convergence of the model

Figure 5.2: This illustration presents five distinct architectures that were tested for the model. Ultimately, model 3 was chosen as the selected architecture. Parentheses contain the data dimensions following each operation. The application of a dropout layer is highlighted in light blue text.

can be assessed by taking notice that the model's performance metrics have stabilised and do not show big improvements with additional epochs, while overfitting is apparent when the model performs well on training data but poorly on unseen data. Evaluation metrics such as accuracy, recall, specificity, false positive rate, precision, and F1-score were used to assess model performance. To evaluate the model's success, the values of the recall, precision, and F1-score were prioritised, aiming for values approaching 1, which are indicative of high performance.
Following the selection of the architecture, two different experiments were conducted:

- Evaluating the model's classification capabilities;
- Comparing a model with the original Matlab tool.

For evaluating the model's classification capabilities, the dataset was split into three subsets randomly selected, where roughly 80% of the dataset was used for training, 10% was used for validation, and 10% was used for testing. The selection of each subset followed the same criteria as when selecting the best architecture, where an independent folder containing multiple, previously unseen, .wav files composed the testing subset.
Three separate scenarios were created to evaluate the model's classification capabilities under different conditions. These resulted in models trained and tested using 2-second spectrogram windows that were designed to contain at least one click, with a high probability of containing multiple clicks, or represent a 'no-click' class if devoid of clicks entirely. Table 5.1 shows all the models created for each scenario: $Md0$ and $Zc0$ compose scenario 1, $Md1$ and $Zc1$ compose scenario 2, and $Bw0$ composes scenario 3.
Scenario 1 corresponds to a binary classifier of presence and absence events. Model $Md0$, present in scenario 1, was composed by an independent dataset totaling 3.4 hours of audio, which corresponded to a total of 6065 spectrograms (1405 spectrograms where clicks from Blainville's beaked whales were present, and 4660 spectrograms with no clicks present). Model $Zc0$, also present in scenario 1, was composed by an independent dataset totaling 15.8 hours of audio, which corresponded to a total of 28370 spectrograms (6111 spectrograms where clicks from Cuvier's beaked whales were present, and 22259 spectrograms with no clicks present). The aim of the models present in scenario 1 was to correctly detect the targeted species clicks, Blainville's in the case of $Md0$ and Cuvier's in the case of $Zc0$, and consider all the spectrograms with no clicks as noise.
Scenario 2 also represents a binary classifier for presence and absence events. However, in this case, spectrograms from non-targeted beaked whale species, as well as spectrograms without clicks, are grouped into a single class and treated as noise. In this scenario, the goal is to confuse the model by introducing not only spectrograms without clicks but also vocalisations similar to those of the targeted species, in this case with another beaked whale species, in order to see if the model will nonetheless correctly identify the events of the targeted species. Although this scenario is not going to occur in practice, given that the locations of the tagged Blainville's

and Cuvier's beaked whales were distinct, creating a model in which a similar species was added to confuse it while still managing to distinguish between the targeted and non-targeted beaked whales could be helpful in a scenario in which some additional noise corresponding to another species was captured by the tagged whale, making the model more robust. Model $Md1$, present in scenario 2, was composed by the same independent dataset applied in $Md0$, and model $Zc1$ by the one applied in $Zc0$. The aim of the models present in scenario 2 was to correctly detect the targeted species clicks, Blainville's in the case of $Md1$, and Cuvier's in the case of $Zc1$, considering the spectrograms of the non-targeted beaked whale species clicks and the spectrograms with no clicks as a single noise class.
Scenario 3 corresponds to a three-class classifier. Model $Bw0$, present in this scenario, was composed by an independent dataset totaling 19.2 hours of audio, which corresponded to a total of 334435 spectrograms (1405 spectrograms where clicks from Blainville's beaked whales were present, 6111 spectrograms where clicks from Cuvier's beaked whales were present, and 26919 spectrograms with no clicks present). The aim of this model was to correctly distinguish vocalisations between two species, in this case, Blainville's from Cuvier's clicks, and spectrograms containing no clicks, which were considered noise.

Table 5.1: Table illustrating all the different models created for testing 3 different scenarios. The parenthesis in models $Md1$ and $Zc1$ indicate that the classes inside them were combined into a single class that is classified as noise. Nc='no-click' class; Md='Blainville's beaked whale clicks' class; Zc= 'Cuvier's beaked whale clicks' class

Model	*composition*	*dataset*	*spectrograms present in testing*	*model's goal*
$Md0$	Presence: Md, Absence: Nc (binary classifier)	3 .wav files (3.4 hours total)	Total: 6065 (1405 presence, 4660 absence)	detect Md clicks, consider Nc as noise
$Md1$	Presence: Md, Absence: (Nc+Zc) (binary classifier)		Total: 6065 (1405 presence, 4660 absence)	detect Md clicks, consider (Nc+Zc) as noise
$Zc0$	Presence: Zc, Absence: Nc (binary classifier)	12 .wav files (15.8 hours total)	Total: 28370 (6111 presence, 22259 absence)	detect Zc clicks, consider Nc as noise
$Zc1$	Presence: Zc, Absence: (Nc+Md) (binary classifier)		Total: 28370 (6111 presence, 22259 absence)	detect Zc clicks, consider (Nc+Md) as noise
$Bw0$	Md + Zc + Nc (3 class classifier)	15 .wav files (19.2 hours total)	Total: 34435 (1405 Md, 6111 Zc, 26919 Nc)	distinguish Zc clicks from Md clicks, consider Nc as noise

For comparing the CNN models with the original Matlab tool, the dataset was split into three subsets, namely, training, validation, and testing. However, the testing subset was not randomly selected. To allow for a proper comparison with the Matlab tool, the model testing datasets were the same as those personally annotated in the tool. Since the Matlab tool analysed every click present in the dataset provided, different spectrogram windows from the ones applied for evaluating the model's classification capabilities had to be used. Consequently, models were trained and tested using 0.08-second spectrogram windows. These windows were designed to have a high probability of containing one click per window, given the inter-click interval of Blainville's beaked whales, or represent a 'no-click' class if devoid of clicks entirely. Two different scenarios were created to compare the CNN models developed with the original Matlab tool.

Table 5.2 illustrates the CNN models created for each scenario.

Scenario 1 was composed of two test experiments: *binary test* 1 and *binary test* 2. These test experiments correspond to a single binary classifier model of presence and absence events that was then tested with two distinct datasets. *Binary test* 1 was composed by an independent dataset, *test set* 1, totaling 17.8 hours of audio, which corresponded to a total of 802452 spectrograms (26148 spectrograms where clicks from Blainville's beaked whales were present, and 776304 spectrograms with no clicks present). *Binary test* 2 was composed by an independent dataset, *test set* 2, totaling 18.9 hours of audio, which corresponded to a total of 849688 spectrograms (36055 spectrograms where clicks from Blainville's beaked whales were present, and 813633 spectrograms with no clicks present). The goal of the model for both of these test experiments was to correctly detect Blainville's beaked whale clicks and classify all the spectrograms with no clicks as noise.

Scenario 2 was composed of test experiments: *three class test* 1 and *three class test* 2. These test experiments correspond to a single three-class classifier model that was then tested with two distinct datasets. *Three class test* 1 was composed by the same testing dataset as *binary test* 1; however, in this case, instead of just considering a presence and absence class, three classes were considered. A total of 802452 spectrograms (26148 spectrograms where clicks from Blainville's beaked whales were present, 0 spectrograms where clicks from Cuvier's beaked whales were present, and 776304 spectrograms with no clicks present). *Three class test* 2 was composed by the same testing dataset as *binary test* 2, and, similarly to *three class test* 1, also considered 3 classes. There were a total of 849688 spectrograms (36055 spectrograms where clicks from Blainville's beaked whales were present, 0 spectrograms where clicks from Cuvier's beaked whales were present, and 813633 spectrograms with no clicks present). The goal of the model for both test experiments was to correctly distinguish between the vocalisations of two species, specifically Blainville's and Cuvier's beaked whales, and to classify spectrograms containing no clicks as noise. Although no Cuvier's clicks were present in test sets 1 and 2, the inclusion of Cuvier's beaked whale clicks in the training dataset was important. This ensured that the model would need to be capable of identifying the absence of Cuvier's clicks while analysing each spectrogram window of the test sets, which in turn would improve the robustness of the model in distinguishing Blainville's and Cuvier's beaked whale clicks. Once these models were tested, they were compared with the Matlab tool by analysing the performance metrics taken from the confusion matrices.

The methodology, involving the training and testing of the models used in both experiments is represented in Figure 5.3.

Table 5.2: Table illustrating the CNN models created to compare with the Matlab original tool. Nc=‘no-click’ class; Md=‘Blainville’s beaked whale clicks’ class; Zc= ‘Cuvier’s beaked whale clicks’ class

Test experiment ID	*composition*	*dataset*	*spectrograms present in testing*	*model's goal*
Binary test 1	Presence: Md, Absence: Nc (binary classifier)	test set 1 (19 .wav files, 17.8 hours total)	Total: 802452 (26148 presence, 776304 absence)	detect Md clicks, consider Nc as noise
Binary test 2	Presence: Md, Absence: Nc (binary classifier)	test set 2 (20 .wav files, 18.9 hours total)	Total: 849688 (36055 presence, 813633 absence)	detect Md clicks, consider Nc as noise
Three class test 1	Md + Zc + Nc (3 class classifier)	test set 1 (same as above)	Total: 802452 (26148 Md, 0 Zc, 776304 Nc)	distinguish Md clicks from Zc clicks, consider Nc as noise
Three class test 2	Md + Zc + Nc (3 class clssifier)	test set 2 (same as above)	Total: 849688 (36055 Md, 0 Zc, 813633 Nc)	distinguish Md clicks from Zc clicks, consider Nc as noise

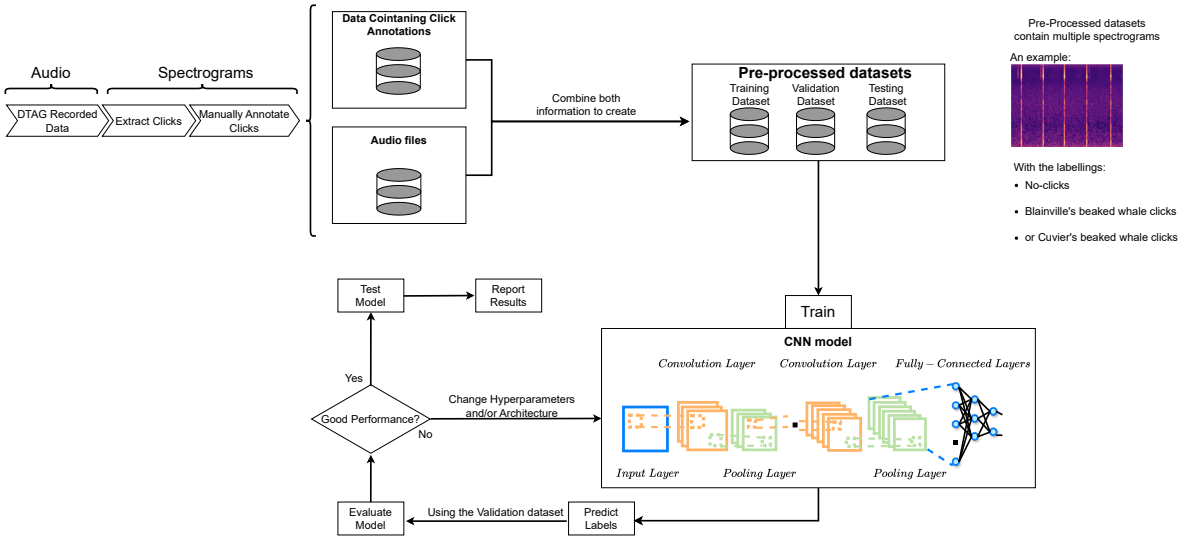


Figure 5.3: Overview of the model’s pipeline, delineating the operational functionality and step-by-step utilisation of the utilised CNN model for species and click classification within this project.

5.5 Implementation

This section provides an in-depth overview of the hardware, software, and hyperparameters that were taken into account during the development of the CNN model.

5.5.1 Hardware and software

The original manual annotations were performed in Matlab R2019a using Mark Johnson’s ‘d3matlab’ tool where all functions applied were present, with a parallel use of Sonic Visualiser v.4.5.2 for audio file playback during annotation. Subsequently, the spectrograms provided to the model were generated using the Librosa library v.0.10.0 ([93]) in Python v3.9.6. The model was developed using Tensorflow package v.2.10.0 ([27]) and Keras v.2.10.0 ([1], both utilised

within Python. Training of the models occurred on a computer equipped with an SSD M.2 2280 Kingston NV2 2TB 3D QLC NVMe PCIe Gen 4.0x4, 32GB RAM, an EVGA GeForce RTX 3060 XC GPU, and a 13th Gen Intel Core i5-13600KF CPU at 3.50 GHz.

5.5.2 Hyperparameters

For this thesis problem, the CNN model was trained for 20 epochs. This choice was made because the model's learning curve stopped showing significant improvements after about 20 epochs. It was a balance between spending reasonable time on training and getting better performance from the model. The decision to use the Adam optimiser was based on its successful application in similar ecological studies ([139, 29, 14]).
The final model's hyperparameters were influenced by two studies ([29, 14]). These hyperparameters, include the number of layers, the quantity of filters in each convolutional layer, the use of dropout layers, the type of activation function chosen, and the optimiser along with its learning rate applied to the model. The best model emerged as a blend of the hyperparameters found in both studies. This model comprised two convolution layers followed by two max-pooling layers. The number of filters was set to 32 and 50 in the two convolution layers respectively. The increase in filters in the second layer aimed to capture more intricate features and enhance the model's ability to understand complex relationships within the data. The most effective model also featured a dropout layer set to 0.25 and utilised a sigmoid activation function. The chosen optimiser was Adam with a learning rate of 0.001.

Chapter 6

Results and Discussion

6.1 Introduction

The aim of this chapter is to provide the results obtained throughout the project. The primary focus was to create a model that could detect clicks from both Blainville's and Cuvier's beaked whales. Based on the performance metrics indicated in section 5.3, the approaches used for each model are going to be assessed. The first section focuses on model classification performance, while the final section compares the model's performance with the original Matlab tool.

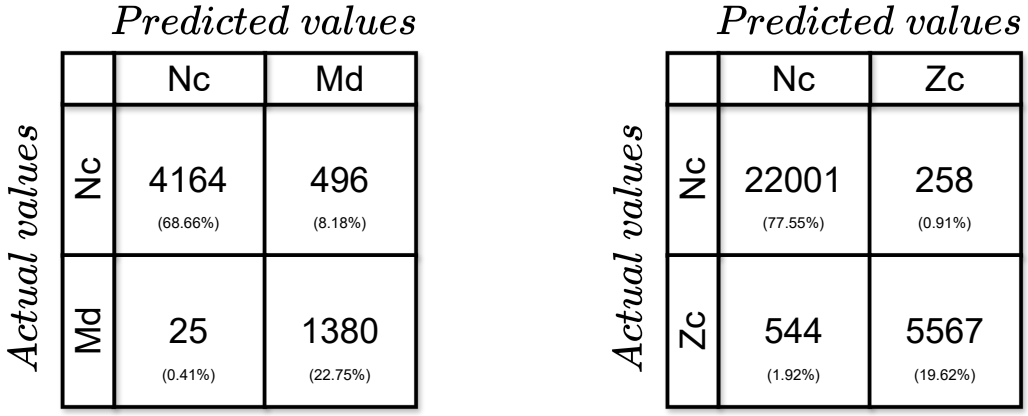
6.2 Model's Classification Capabilities

The evaluation of the CNN model's classification capability relies on multiple performance metrics, particularly confusion matrices and values derived from them such as accuracy, recall, false positive rate, specificity, precision and F1-score. Due to the nature of the data collected some classes were disproportionately represented, resulting in an imbalanced distribution across them. Therefore, confusion matrices were considered important to summarise, through a table, and visualise the ground truth labels versus the model's predictions. These models and their respective findings will be presented and discussed throughout this chapter.

6.2.1 Scenario 1

Initially, establishing a model with the capacity to discriminate between noise and the various click patterns associated with Blainville's and Cuvier's beaked whales was the main objective. To test this hypothesis, scenario 1 was created, where two distinct models corresponding to a binary classifier of presence and absence events were built, the model *Md0* and the model *Zc0*, presented in table 5.1.

The confusion matrices corresponding to both models present in this scenario are illustrated in figure 6.1, from which the following is observed:
In model *Md0*, 1380 spectrograms were correctly identified as presence events (true positives), and 4167 spectrograms without clicks were correctly identified as absence events (true negatives). However, in model *Md0* 496 spectrograms without clicks were incorrectly identified as clicks (false positives), and 25 spectrograms with clicks were incorrectly identified as absence events (false negatives).
In model *Zc0*, there were 5567 spectrograms correctly identified as presence events, indicating the true positives, and 22001 spectrograms without clicks were correctly identified as absence events (true negatives). Still, in model *Zc0* there were 258 spectrograms without clicks were incorrectly identified as clicks (false positives), and 544 spectrograms with clicks were incorrectly identified as absence events (false negatives).



(a) Confusion Matrix of Blainville's beaked whale (b) Confusion Matrix of Cuvier's beaked whale

Figure 6.1: Confusion matrices of Blainville's beaked whale (model Md0), and Cuvier's beaked whale (model Zc0) demonstrating the capacity of the models to discriminate between clicks and noise. Nc='no-click' class; Md='Blainville's beaked whale clicks' class; Zc= 'Cuvier's beaked whale clicks' class.

Using these values, it is possible to calculate the performance metrics illustrated in table 6.1. This table shows that model *Md0* successfully detected calls with an accuracy of 91.41%, whereas model *Zc0* successfully detected calls with an accuracy of 97.17%. Even though these high accuracies are important, they should be understood alongside metrics like precision, recall, and F1-score due to the class imbalance present in both models, which is indicated by the percentages of each class in the confusion matrices. Taking these metrics into account, model

*Md*0 had a recall value of 98.22%, showing that the model is effective at detecting the presence of Blainville's beaked whale clicks, missing only a small number of actual clicks. The precision value of model *Md*0 corresponded to 73.56%. This is indicative that while the model is effective at detecting clicks, there are still a number of instances where it incorrectly classified absence events as presence ones. The value of the F1-score of this model was 84.12%.
In the case of model *Zc*0, its recall value was 91.10%, also showcasing its effectiveness in detecting the presence of Cuvier's beaked whale clicks. The precision value of model *Zc*0 was 95.57%, which is indicative that is effective at identifying clicks and has a low rate of false positives. The value of the F1-score of this model was 93.28%.
The appendix section 7.2 presents examples of spectrograms for both Blainville's and Cuvier's beaked whales, illustrating cases of true positives, false positives, true negatives, and false negatives.

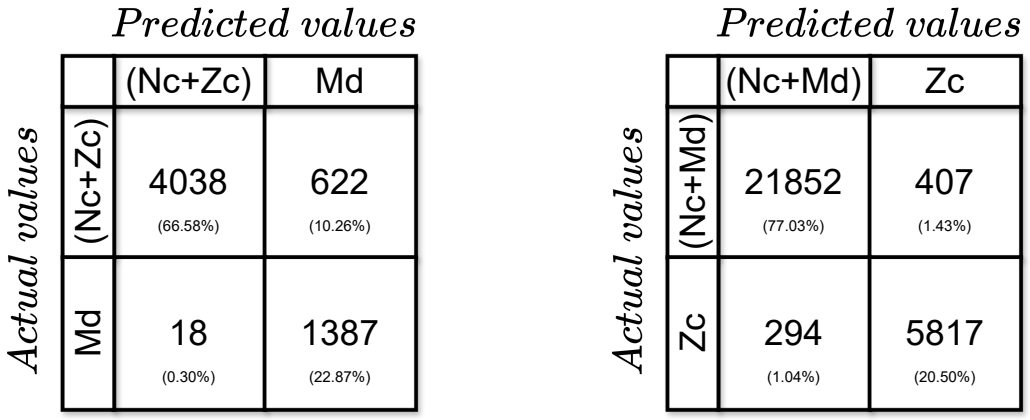
Table 6.1: Performance metrics of models *Md*0 and *Zc*0.

Metrics	*Md*0	*Zc*0
Accuracy	91.41%	97.17%
Recall	98.22%	91.10%
Specificity	89.36%	98.84%
False Positive Rate	10.64%	1.16%
Precision	73.56%	95.57%
F1-score	84.12%	93.28%

6.2.2 Scenario 2

In this scenario, already mentioned in Section 5.4, a binary classifier for presence and absence events was created. However, what sets scenario 2 apart from the previous one is the fact that spectrograms from non-targeted beaked whale species and spectrograms without clicks were grouped into a single class, considered noise, while spectrograms from the targeted beaked whale species were placed in a separate class. In order to test scenario 2, two models were created, namely *Md*1 and *Zc*1, present in table 5.1. The confusion matrices of both models are depicted in figure 6.2, from which the following is observed:
In model *Md1*, 1387 spectrograms were correctly identified as presence events (true positives), and 4038 spectrograms without clicks were correctly identified as absence events (true negatives). In contrast, in model *Md1* 622 spectrograms without clicks were incorrectly identified as clicks (false positives), and 18 spectrograms with clicks were incorrectly identified as absence events (false negatives).
In model *Zc1*, there were 5817 spectrograms correctly identified as presence events, indicating the true positives, and 21852 spectrograms without clicks were correctly identified as absence events

(true negatives). Yet, in model *Zc1* there were 407 spectrograms without clicks were incorrectly identified as clicks (false positives), and 294 spectrograms with clicks were incorrectly identified as absence events (false negatives).



(a) Confusion Matrix of Blainville's beaked whale (b) Confusion Matrix of Cuvier's beaked whale

Figure 6.2: Confusion matrices of Blainville's beaked whale (model Md1), and Cuvier's beaked whale (model Zc1) demonstrating the capacity of the models to discriminate between presence and absence events. In scenario 2, the absence events correspond to the grouping of the non-targeted species spectrograms with the spectrograms without clicks, which is represented by the parentheses. Nc='no-click' class; Md='Blainville's beaked whale clicks' class; Zc= 'Cuvier's beaked whale clicks' class

Table 6.2: Performance metrics of models $Md1$ and $Zc1$.

Metrics	$Md1$	$Zc1$
Accuracy	89.45%	97.53%
Recall	98.72%	95.19%
Specificity	86.65%	98.17%
False Positive Rate	13.35%	1.83%
Precision	69.04%	93.46%
F1-score	81.25%	94.32%

Table 6.2 shows the different metrics calculated from the values of the confusion matrices in figure 6.2. Model $Md1$ successfully detected calls with an accuracy of 98.45%, while model $Zc1$ successfully detected calls with an accuracy of 97.53 %. The recall value of model $Md1$ was

98.72%, demonstrating its effectiveness in detecting Blainville's beaked whale clicks from the total number of presence events. Its precision value was 69.04%, showing that although the model is effective at detecting Blainville's beaked whale clicks, there are a considerable number of false positive events. The value of the F1-score of this model was 81.25%. Focusing now on the metrics of model $Zc1$, its recall value was 95.19%, demonstrating its effectiveness in detecting Cuvier's beaked whale clicks from the total number of presence events. The precision value of $Zc0$ was 93.46%, which is indicative that is effective at identifying clicks and has a low rate of false positives. The value of the F1-score of this model was 94.32%.

6.2.3 Scenario 3

Scenario 3 constituted a three-class classifier model, present in table 5.1, where its aim was to correctly distinguish vocalisations between two species, Blainville's from Cuvier's beaked whale clicks, and spectrograms containing no clicks, which were considered noise. The confusion matrix illustrated in figure 6.4 presents all the classification results for each class by the model $Bw0$, as well as their distributions in percentage terms. From this confusion matrix the following is observed:
In the first row of the confusion matrix, 25672 spectrograms without clicks were correctly identified (Nc), 994 spectrograms without clicks were misclassified as Blainville's beaked whale clicks (Md), and 253 spectrograms without clicks were misclassified as Cuvier's beaked whale clicks (Zc). In the second row, 17 spectrograms containing Blainville's beaked whale clicks were misclassified as instances of noise (Nc), 1386 spectrograms containing Blainville's beaked whale clicks were correctly identified (Md), and 2 spectrograms containing Blainville's beaked whale clicks were misclassified as Cuvier's beaked whale clicks (Zc). In the third row, 402 spectrograms containing Cuvier's beaked whale clicks were misclassified as instances of noise (Nc), 5 spectrograms containing Cuvier's beaked were misclassified as Blainville's beaked whale clicks (Md), and 5704 spectrograms containing Cuvier's beaked whale clicks were correctly identified (Zc).
Taking into consideration the values from the confusion matrix above, the performance metrics illustrated in table 6.3 were calculated. This table shows that model $Bw0$ successfully detected Blainville's calls with an accuracy of 97.04%, Cuvier's calls with an accuracy of 98.08%, and spectrograms without clicks, considered noise, with an accuracy of 95.16%. The recall value for Blainville's calls was 98.65%, for Cuvier's it was 93.34%, and for the noise was 95.37%. These values demonstrate the model's effectiveness at measuring the proportion of events correctly identified for each class. The precision value of model $Bw0$ for Blainville's calls corresponded to 58.11%, for Cuvier's to 95.72%, and for the noise to 98.39%. These values show that, while the model was effective at identifying clicks with a low rate of false positives in the case of Cuvier's calls and noise, when it came to Blainville's calls, the model classified them incorrectly a considerable number of times, presenting a performance close to a random classification (around

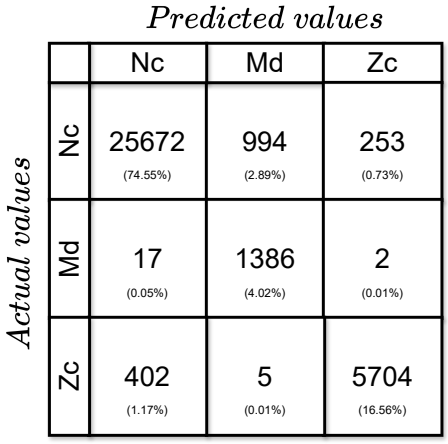


Figure 6.3: Confusion Matrix of Beaked whales

Figure 6.4: Confusion matrices of model $Bw0$, showcasing a three-class classifier whose goal is to distinguish Zc clicks from Md clicks and consider Nc as noise. Nc='no-click' class; Md='Blainville's beaked whale clicks' class; Zc= 'Cuvier's beaked whale clicks' class.

50%). The F1-score values of this model were 73.14%, regarding the Blainville's calls, 94.51%, regarding the Cuvier's, and 96.86%, regarding the noise.

6.2.4 Discussion of the results of Model's Classification Capabilities

Upon examining the metrics in tables 6.1, 6.2, and 6.3, there is a clear difference in the models' performance between the two species. The models $Zc0$, $Zc1$, and $Bw0$ for the Cuvier's beaked whale demonstrated high levels of performance, with high recall and precision values, resulting in high F1-scores. This indicates that the models are very efficient in detecting all relevant instances (true positives) while minimising the identification of irrelevant instances (false positives), thus making them highly effective at detecting Cuvier's beaked whale clicks. In contrast, the models $Md0$, $Md1$, and $Bw0$ for Blainville's beaked whales exhibited a trade-off between precision and recall. Although these models are effective at detecting actual clicks (high recall), they also present several instances of false positives (poor precision). This means that the high recall seems to negatively impact the precision, leading to a higher number of irrelevant instances being identified as presence events. In order to improve the models for the Blainville's beaked whale it would still be needed to find a balance between these two metrics, that would consequently improve the overall F1-score as well.
While the models' differing performance between the species is evident, the underlying reasons

Table 6.3: Performance metrics of model *Bw*0.

Metrics	*Bw*0
Accuracy_{noise}	95.16%
Accuracy_{md}	97.04%
Accuracy_{zc}	98.08%
Recall_{noise}	95.37%
Recall_{md}	98.65%
Recall_{zc}	93.34%
$\text{Specificity}_{noise}$	94.43%
Specificity_{md}	96.98%
Specificity_{zc}	99.10%
$\text{False Positive Rate}_{noise}$	5.57%
$\text{False Positive Rate}_{md}$	3.02%
$\text{False Positive Rate}_{zc}$	0.90%
Precision_{noise}	98.39%
Precision_{md}	58.11%
Precision_{zc}	95.72%
F1-score_{noise}	96.86%
F1-score_{md}	73.14%
F1-score_{zc}	94.51%

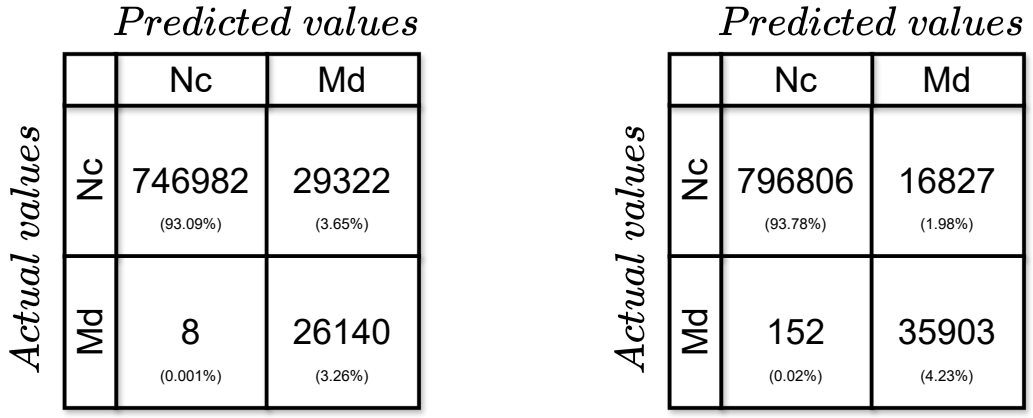
for this disparity remain somewhat unclear. Several factors should be considered when interpreting these results. First, it is important to note that annotations for Blainville's beaked whale clicks were done manually, whereas annotations for Cuvier's beaked whale clicks were created by other researchers and provided through the ACCURATE project. Although the manual annotations were made with great care, the potential for human error cannot be completely disregarded, as it may impact model performance. Additionally, the recordings for the two species were collected from different locations, possibly introducing varying levels of background noise. This variability in recording environments may have contributed to the lower precision observed for the Blainville's beaked whale models, as high levels of noise tended to increase false positive classifications. However, the reason why this same susceptibility to false positives under noisy conditions is not observed in the Cuvier's beaked whale models remains unknown and calls for a deeper investigation in future studies. Moreover, it should also be considered that the amount of data used to train these models is relatively small by deep learning standards. To build more robust models with better generalisation, an increase in both data volume and annotations would be essential. This would allow for a dataset that is more representative of real world conditions, with a broader range of noise profiles and click types, ultimately resulting in models that are more resilient and adaptable to diverse acoustic environments. Future work should focus on analysing these systematically.

6.3 CNN model comparison with Matlab original tool

To evaluate and compare the classification capabilities of the CNN model with the original Matlab tool for detecting clicks in DTAG audio files, two separate models were built. As noted in section 5.4, the testing subset was not chosen randomly to evaluate the model on the same files annotated in the tool. Table 5.2 depicts two scenarios, each with two test experiments. Similar to section 6.2, confusion matrices were used to display the ground truth labels versus the model's predictions. Performance metrics were calculated based on the matrix values and compared to Matlab tool performance metrics.

6.3.1 Scenario 1

Scenario 1 corresponded to a binary classifier model of presence and absence events tested with two distinct datasets (section 5.4). This resulted in two test experiments: *Binary test 1* and *Binary test 2*, described in table 5.2. The confusion matrices corresponding to both test experiments present in this scenario are illustrated in figure 6.5, from which the following is observed:
In experiment *Binary test 1*, 26140 spectrograms were correctly identified as presence events (true positives), and 746982 spectrograms without clicks were correctly identified as absence events (true negatives). However, in experiment *Binary test 1* 29322 spectrograms without clicks were incorrectly identified as clicks (false positives), and 8 spectrograms with clicks were incorrectly identified as absence events (false negatives).
In experiment *Binary test 2*, there were 35903 spectrograms correctly identified as presence events (true positives), and 796806 spectrograms without clicks were correctly identified as absence events (true negatives). In contrast, in experiment *Binary test 2* there were 16827 spectrograms without clicks were incorrectly identified as clicks (false positives), and 152 spectrograms with clicks were incorrectly identified as absence events (false negatives).
Taking the values of both confusion matrices into consideration, it is possible to calculate the metrics and compare them with the Matlab tool ones, as illustrated in table 6.4. Metrics of both the *CNN* model and the *Matlab* tool evaluated on dataset '*test set* 1' presented several metrics with similar results, showing a high accuracy of 96.34%, and 98.34%, for *binary test* 1 and the *Matlab* tool, respectively. They also presented high recall values, where the *binary test* 1 was 99.97%, and the *Matlab* tool was 98.77%. However, the clear distinction in performance between the two was in the precision metrics; the *binary test* 1 presented a value of only 47.13%, while the *Matlab* tool had a value of 97.91%. This indicates that the CNN model on test set 1 identified Blainville's calls with a high percentage of false positives. The significant difference in the precision metric values between the two also contributed to a considerable difference in the F1-score values. *Binary test* 1 had an F1-score of 64.06%, and the *Matlab* tool had an F1-score of 98.34%. These metrics reveal that on test set 1, the *Matlab* tool presented a better overall



(a) Confusion Matrix of Binary test 1

(b) Confusion Matrix of Binary test 2

Figure 6.5: Confusion matrices of Blainville's beaked whale, corresponding to test experiment ID: Binary test 1, and Binary test 2. Nc='no-click' class; Md='Blainville's beaked whale clicks' class.

performance. When comparing the performance metrics on the '*test set* 2' dataset, the *CNN* model performs much better than the *Matlab* tool, apart from the precision metric, where both present similar values. *Binary test* 2 detected Blainville's beaked whale calls with an accuracy of 98.00%, which was significantly better than the 67.73% presented by the *Matlab* tool. The recall value of *binary test* 2 was 99.58%, which is also much better than the 66.62% of the *Matlab* tool. The precision was the metric with the most similar values, namely, 68.09% for the *binary test* 2, and 68.87% for the *Matlab*. This resulted in an F1-score of 80.88% for *binary test* 2 and of 67.73% for the *Matlab* tool, revealing a better overall performance of the *CNN* model on *test set* 2.

6.3.2 Scenario 2

In scenario 2 of the CNN model comparison with the original MATLAB tool experiment, a three-class classifier model, mentioned in section 5.4, was developed to distinguish vocalisations among three categories: Blainville's beaked whale clicks, Cuvier's beaked whale clicks, and spectrograms containing no clicks (considered as noise). This model was divided into two test experiments, namely, *Three class test* 1 and *Three class test* 2, shown in table 5.2. The confusion matrices corresponding to both test experiments present in this scenario are illustrated in figure 6.6, from which the following is observed:

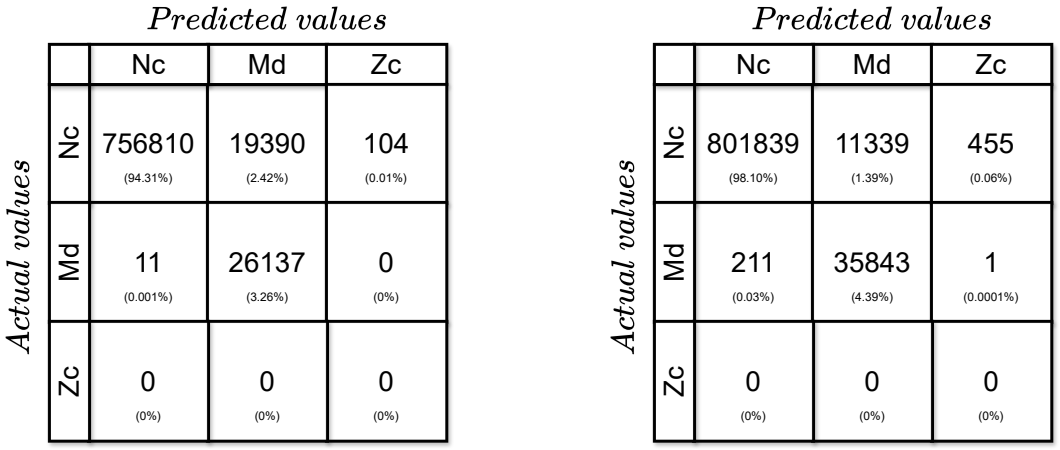
In the first row of the confusion matrix, in *Three class test* 1, 756810 spectrograms without clicks

Table 6.4: Comparison between the performance metrics of CNN models 'binary test 1' and 'binary test 2' with Matlab tool for test set 1 and test set 2.

	test set 1		*test set* 2	
Metrics	*CNN*(*binary test* 1)	*Matlab*	*CNN*(*binary test* 2)	*Matlab*
Accuracy	96.34%	98.34%	98.00%	67.73%
Recall	99.97%	98.77%	99.58%	66.62%
Specificity	96.22%	97.91%	97.93%	68.87%
False Positive Rate	3.78%	2.09%	2.07%	31.13%
Precision	47.13%	97.91%	68.09%	68.87%
F1-score	64.06%	98.34%	80.88%	67.73%

were correctly identified (Nc), 19390 spectrograms without clicks were misclassified as Blainville's beaked whale clicks (Md), and 104 spectrograms without clicks were misclassified as Cuvier's beaked whale clicks (Zc). While in *Three class test* 2, 801839 spectrograms without clicks were correctly identified (Nc), 11339 spectrograms without clicks were misclassified as Blainville's beaked whale clicks (Md), and 455 spectrograms without clicks were misclassified as Cuvier's beaked whale clicks (Zc). In the second row, in *Three class test* 1, 11 spectrograms containing Blainville's beaked whale clicks were misclassified as instances of noise (Nc), 26137 spectrograms containing Blainville's beaked whale clicks were correctly identified (Md), and 0 spectrograms containing Blainville's beaked whale clicks were misclassified as Cuvier's beaked whale clicks (Zc). While in *Three class test* 2, 211 spectrograms containing Blainville's beaked whale clicks were misclassified as instances of noise (Nc), 35843 spectrograms containing Blainville's beaked whale clicks were correctly identified (Md), and 1 spectrograms containing Blainville's beaked whale clicks were misclassified as Cuvier's beaked whale clicks (Zc). In the third row, both in *Three class test* 1 and *Three class test* 2, no spectrograms classifications were done.

Table 6.5 displays the different metrics calculated based on the values of the confusion matrices present in figure 6.6. Identical to the results observed in table 6.4, for the binary classifier model, the metrics of the *CNN* model and the *Matlab* tool evaluated on dataset '*test set* 1', for the three-class classifier model, were once again, in most cases, similar. The *CNN* model *three class test* 1 successfully detected calls with an accuracy of 97.58%, while the *Matlab* tool successfully detected calls with an accuracy of 98.34%. The recall value of *three class test* 1 was 99.96%, slightly better than the value of the *Matlab* tool, which was 98.77%. When comparing the precision metrics of the three-class classifier model to those of the binary classifier model, it appears that adding an extra class during training, Cuvier's beaked whale clicks, which are similar to the target species, had a positive effect on the reduction in the percentage of false positive detections while identifying Blainville's calls, achieving a metric of 57.41%. Nonetheless, the *Matlab* tool presented a much better precision value of 97.91%. This resulted in an F1-



(a) Confusion Matrix of Three class test 1

(b) Confusion Matrix of Three class test 2

Figure 6.6: Confusion matrices of Blainville's beaked whale, corresponding to test experiment ID: Three class test 1, and Three class test 2. Nc='no-click' class; Md='Blainville's beaked whale clicks' class; Zc='Cuvier's beaked whale clicks' class.

score of 72.93% for the *three class test* 1 and of 98.34% for the *Matlab* tool, which once again indicates the overall better performance of the *Matlab* tool over the *CNN* model on the dataset '*test set* 1'. Performance metrics on dataset '*test set* 2', show that, once more, the CNN model, *three class test* 2, outperformed the *Matlab* tool. The CNN model detected Blainville's beaked whale calls with an accuracy of 98.64%, which is significantly higher than the *Matlab* tool's 67.73%. The recall rate for the *CNN* model was 99.41%, which was greater than the *Matlab* tool's 66.62%. Precision, for *three class test* 2, was far better than the precision value of the *binary test* 2, with the *CNN* model value of 75.97%, reinforcing the idea that by adding a class with similar characteristics to the target class in training, the model gained robustness, and the *Matlab* tool value was 68.87%. As a result, the *CNN* model got an F1-score of 86.12%, compared to 67.73% for the *Matlab* tool, indicating that the CNN model performed better overall on '*test set* 2'.

6.3.3 Discussion of the results of the CNN model comparison with Matlab original tool

Considering the limitations of the models discussed in sub-section 6.2.4 when evaluating Blainville's beaked whale, and given that section 6.3 focuses solely on models dealing with Blainville's beaked whales, it is evident that higher recall values in this section have resulted in lower precision val-

Table 6.5: Comparison between the performance metrics of CNN models 'three class test 1' and 'three class test 2' with Matlab tool for test set 1 and test set 2.

	test set 1		*test set* 2	
Metrics	*CNN*(*three class test* 1)	*Matlab*	*CNN*(*three class test* 2)	*Matlab*
Accuracy_{md}	97.58%	98.34%	98.64%	67.73%
Recall_{md}	99.96%	98.77%	99.41%	66.62%
Specificity_{md}	97.50%	97.91%	98.61%	68.87%
$\text{False Positive Rate}_{md}$	2.50%	2.09%	1.39%	31.13%
Precision_{md}	57.41%	97.91%	75.97%	68.87%
F1-score_{md}	72.93%	98.34%	86.12%	67.73%

ues. This outcome is likely because each spectrogram window represents only 0.08 seconds, as opposed to the 2-second windows used in section 6.2. Consequently, the models were able to correctly identify actual clicks more frequently but also confused noise with actual clicks more often (resulting in lower precision). This could possibly be due to events where there is some similarity between noise patterns and the clicks or due to the models being overly sensitive in detecting clicks (high recall). This leads to models overpredicting clicks, which results in many false positives.

The imbalance between classes in the testing dataset is another aspect that leads to the low precision values and should be mentioned. There are far more absence events than presence events, with noise making up over 90% of the data in section 6.3 models and over 70% in section 6.2 models. This imbalance affects performance evaluation, particularly in the precision metric. Since, for section 6.3 models, there are approximately nine absence events for every presence event, the models have many more opportunities for false positives, which reduces precision.

Despite all these limitations, table 5.2 shows that, for '*test set* 2', the *CNN* model outperformed the *Matlab* tool. Another important aspect to consider is the time used for data annotation. As mentioned in sub-section 5.2.4, the average time for manual annotation was approximately two and a half hours per hour of audio. Thus, manually annotating the testing datasets used to compare the CNN model with the Matlab tool required between 44 and 47 hours. In contrast, the CNN model inference time took only about 10 minutes, reducing the time needed to analyse the audio files by 99.6%. Given that long-term monitoring will generate thousands of hours of audio across multiple recordings, this reduction in analysis time is significant and should be a factor to consider. However, using the CNN model required some initial work. The data had to be manually labelled at first, and more time was necessary to train the model. These actions are required before the model can be successfully implemented. The true cost savings and efficiency improvements of utilising a CNN model become apparent only once the training is completed and the model performs well. Following these early steps, the more data acquired in the future,

the more advantageous the CNN model gets when compared to human effort.

Chapter 7

Conclusion

This thesis aimed to create and implement a model for automatically classifying beaked whale vocalisations through the implementation of a convolutional neural network, serving as a crucial step towards estimating the density of beaked whales by estimating their cue production rates from animal-borne tags. To do so, several training and testing scenarios were conducted to evaluate the CNN models and assess their classification capabilities under different conditions, resulting in the models present in table 5.1. Three different scenarios were evaluated: one to establish the models' capacity to discriminate between presence and absence events; another to evaluate the model's ability to consistently prioritise a designated target class while considering all the remaining data as noise; and a third to assess the model's ability to distinguish between three different classes. The test results in these different scenarios revealed that the models presented F1-scores between 73.14% and 94.51% but were limited by their lower precision values, where the lowest presented values of 58.11%.

Certain models' limitations must be addressed. Although models $Zc0$, $Zc1$, and $Bw0$ demonstrated a good capacity for detecting Cuvier's beaked whale clicks while classifying the remaining classes as noise, models $Md0$, $Md1$, and $Bw0$ did not perform as well for Blainville's beaked whale due to low precision values, which was an important metric for the problem at hand. This is because these models were designed to be used to estimate population densities of these whales, and thus high precision ensures that population estimates are based on actual whale vocalisations rather than noise or other species, reducing the risk of overestimating the population.

After evaluating the models' capabilities in different scenarios, a distinct experiment was conducted to compare the CNN models with the original Matlab tool. In this case, two different scenarios were created. Scenario 1 was composed of two test experiments: *Binary test* 1 and *Binary test* 2; Scenario 2 was composed of test experiments: *Three class test* 1 and *Three class test* 2. Upon comparing the models with the Matlab tool, it showed that the overall F1-score of the models tested on the dataset '*test set* 1', namely '*Binary test* 1' and '*Three class test* 1', had a significantly worse performance than the *Matlab* tool. However, the overall F1-score of the models

tested on the dataset '*test set* 2', namely '*Binary test* 2' and '*Three class test* 2', had a significantly better performance than the *Matlab* tool.
Although the original purpose was to create a method for automatic click detection on tagged animals using *CNN* models and compare them to the *Matlab* tool, this thesis attempted to go a step farther. Models *Md*0, *Md*1, *Zc*0, and *Zc*1 were developed in order to automate the click count of tagged whales, as well as the *CNN* binary models that were compared with the *Matlab* tool. However, the *Bw*0 and the *CNN* three-class test models go a bit beyond. These models were created with the goal of being capable of distinguishing between multiple species, making them viable methods for passive acoustic monitoring surveys since they can provide information about the presence and behaviour of different species in a given area.
Developing an automatic acoustic classifier is ever more crucial with the huge recordings that present equipment is capable of gathering. This makes manual labelling infeasible, consuming both time, funds, and manpower. As referenced in sub-section 6.3.3, the inference of CNN models saved 99.6% of the time used to do manual annotations. This significant reduction in time highlights the efficiency and effectiveness of CNN models in handling vast amounts of bioacoustic data.

7.1 Thesis learning outcomes

This MSc marked a shift from an undergraduate degree in Biology to a more quantitative, statistical, and computational profile. In the process, a number of new skills were gained, reflecting considerable personal growth. The following concepts and ideas were learnt throughout this thesis:

- Beaked whale acoustics;
- Understood how cue production rates are fundamental for passive acoustic monitoring for density estimation;
- Visualise and interpret spectrograms;
- Annotate sound files for training machine learning models, in particular beaked whale echolocation clicks;
- Create, organise and process large datasets;
- Transform sound into images by making spectrogram using the *Librosa* library in *Python*;
- *Python* programming;
- Work with *Matlab* scripts and to plot graphs;

- Use '*draw.io*' to create diagrams;
- Use *JabRef* to create and organise bibliographies, and generating references using *BibTex*;
- Write *LaTeX* documents in *Overleaf*;
- Fundamentals of machine learning and different fields within it;
- Work with Machine learning libraries such as *Keras* and *TensorFlow*;
- Create, train, validate and test CNN models in *Python*;

7.2 Future directions

Through this thesis, significant progress has been made in the research to automatically classify beaked whale vocalisations using convolutional neural networks. However, there are opportunities for future research.

It would be important to try to expand the size of the dataset with more annotated data and even try to use some advanced data augmentation techniques to address the issue of class imbalance in beaked whale vocalisation recordings, which could potentially increase the robustness of the CNN models.

The continuing study of more advanced CNN architectures and how their implementation affects the model's performance should also be considered. It is also possible to incorporate a temporal aspect into the model by using a different type of neural network, such as recurrent neural networks and Transformer architectures [150] given their success in audio tasks [94].

Furthermore, the idea of reusing CNN models that have already been trained on this dataset should be investigated, given that fine-tuning existing models may result in more efficient training and improved overall performance.

During the course of this thesis, only data from tags was utilised. However, most PAM data is not retrieved from animal-borne tags but instead from towed or fixed hydrophones. According to Jarvis *et al.*, in [57], tag data has the advantage of providing detailed descriptions of the dive cycles and foraging behaviour of the tagged animal. On the other hand, a disadvantage of tag data compared to other PAM methods is the limited quantity of data retrieved, particularly in terms of spatial and temporal coverage, and the inability to provide population-level inferences. For inferences about population trends, animal density, and monitoring, PAM data from towed or fixed hydrophones is more reliable. These hydrophones are capable of non-stop recording vocalisations from multiple species, giving long-term data for research into ecological trends and human impacts. However, PAM data is generally noisy, making it difficult to build automatic detectors based on this data alone. An effective strategy could include applying transfer learning models, such as those presented in this thesis, to PAM data.

Hyperparameter tuning is also something that could be further explored by systematically studying parameters such as learning rates, batch sizes, and network designs, which include, for example, the number and type of layers integrated.
Finally, it would be interesting to work towards integrating the proposed classification model into ongoing research, as that developed under efforts to understand cue production rates from marine mammals, as is the case with the ACCURATE project. Cue production rates are fundamental to convert densities of sounds into density of animals, and methods such as those developed in this thesis will allow such cue rates to be obtained in a more efficient way than what has been the case historically, requiring time and resource demanding manual processing. Ideally, if used together within monitoring programs, automatic detection methods like the ones developed here could have important consequences cascading through into conservation initiatives.

Bibliography

[1] Github: Keras. `https://github.com/keras-team/keras`. Last assessed March 20 2024.

[2] The IUCN Red List of Threatened Species 2020. *Mesoplodon densirostris*. `https://www.iucnredlist.org/species/13244/50364253`. Last assessed April 23, 2020.

[3] WWF (2024) Living planet report 2024 – a system in peril. `https://www.worldwildlife.org/publications/2024-living-planet-report`.

[4] Mahdi Abolghasemi, Babak Abbasi, Toktam Babaei, and Zahra HosseiniFard. How to effectively use machine learning models to predict the solutions for optimization problems: lessons from loss function. *arXiv preprint arXiv:2105.06618*, 2021.

[5] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, page 8308186, 2017.

[6] Patricia Arranz, Natacha Aguilar de Soto, Peter T. Madsen, Alberto Brito, Fernando Bordes, and Mark P. Johnson. Following a foraging fish-finder: Diel habitat use of Blainville's beaked whales revealed by echolocation. *PLOS ONE*, page 0028353, 2011.

[7] Whitlow WL Au and Mardi C Hastings. *Principles of marine bioacoustics*, volume 510. Springer, 2008.

[8] Pierre Baldi and Peter J Sadowski. Understanding dropout. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *NIPS'13: Proceedings of the 26th international conference on neural information processing systems*, volume 2, pages 2814 – 2822. Curran Associates, Inc., 2013.

[9] Jay Barlow. Trackline detection probability for long-diving whales. In *Marine mammal survey and assessment methods*, pages 209–222. CRC Press, 1st edition, 1999.

[10] Jay Barlow. Cetacean abundance in Hawaiian waters estimated from a summer/fall survey in 2002. *Marine Mammal Science*, 22(2):446–464, 2006.

[11] Gustavo Enrique A. P. A. Batista, Ronaldo Cristiano Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, June 2004.

[12] Mark Baumgartner. The distribution of risso's dolphin (*Grampus griseus*) with respect to the physiography of the northern gulf of Mexico. *Marine Mammal Science*, 13(4):537–725, 10 1997.

[13] Christian Bergler, Hendrik Schröter, Rachael Xi Cheng, Volker Barth, Michael Weber, Elmar Nöth, Heribert Hofer, and Andreas Maier. ORCA-SPOT: An automatic killer whale sound detection toolkit using deep learning. *Scientific Reports*, 9(1):10997, jul 2019.

[14] Peter C Bermant, Michael M Bronstein, Robert J Wood, Shane Gero, and David F Gruber. Deep machine learning techniques for the detection and classification of sperm whale bioacoustics. *Scientific reports*, 9:12588, 2019.

[15] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 1st edition, 2006.

[16] Johan Bjorck, Brendan H. Rappazzo, Di Chen, Richard Bernstein, Peter H. Wrege, and Carla P. Gomes. Automatic detection and compression for passive acoustic monitoring of the african forest elephant. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):476–484, jul 2019.

[17] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[18] Lauren B. Buckley, Emily Carrington, Michael E. Dillon, Carlos García-Robledo, Steven B. Roberts, Jill L. Wegrzyn, and Mark C. Urban. Characterizing biological responses to climate variability and extremes to improve biodiversity projections. *PLOS Climate*, 2(6):e0000226, June 2023.

[19] M. Augustine Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25:536–538, 1847.

[20] Kin Wai Cheuk, Hans Anderson, Kat Agres, and Dorien Herremans. nnAudio: an on-the-fly GPU audio to spectrogram conversion toolbox using 1D convolutional neural networks. *IEEE Access*, 8:161981–162003, 2020.

[21] Rene Y Choi, Aaron S Coyner, Jayashree Kalpathy-Cramer, Michael F Chiang, and J Peter Campbell. Introduction to machine learning, neural networks, and deep learning. *Translational Vision Science & Technology*, 9(2):14–14, 2020.

[22] Colin D. MacLeod. *Beaked whales, overview*, pages 80–83. Encyclopedia of Marine Mammals. Academic Press, 3rd edition, 2018.

[23] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198. ACM, sep 2016.

[24] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, aug 1980.

[25] Sourav De, Sandip Dey, Siddhartha Bhattacharyya, and Surbhi Bhatia. *Advanced data mining tools and methods for social computing*. Hybrid Computational Intelligence for Pattern Analysis and Understanding. Academic Press, 1st edition, 2022.

[26] Natacha Aguilar de Soto, Peter T. Madsen, Peter Tyack, Patricia Arranz, Jacobo Marrero, Andrea Fais, Eletta Revelli, and Mark Johnson. No shallow talk: Cryptic strategy in the vocal communication of blainville's beaked whales. *Marine Mammal Science*, 28(2):E75–E92, jul 2011.

[27] TensorFlow Developers. Tensorflow. `https://doi.org/10.5281/zenodo.10126399`, 2023. Last assessed March 20 2024.

[28] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: a deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655. PLMR, 2014.

[29] Emmanuel Dufourq, Ian Durbach, James P Hansford, Amanda Hoepfner, Heidi Ma, Jessica V Bryant, Christina S Stender, Wenyong Li, Zhiwei Liu, Qing Chen, et al. Automated detection of Hainan gibbon calls for passive acoustic monitoring. *Remote Sensing in Ecology and Conservation*, 7(3):475–487, 2021.

[30] Charlotte Dunn, Leigh Hickmott, Darren Talbot, Ian Boyd, and Luke Rendell. Mid-frequency broadband sounds of Blainville's beaked whales. *Bioacoustics*, 22(2):153–163, jun 2013.

[31] Poonguzhali Elangovan and Malaya Kumar Nath. A novel shallow convnet-18 for malaria parasite detection in thin blood smear images. *SN Computer Science*, 2(5):380, 2021.

[32] Christine Erbe, Rebecca Dunlop, and Sarah Dolman. *Effects of noise on marine mammals*, volume 66 of *Springer Handbook of Auditory Research*, chapter 10, pages 277–309. Springer, New York, NY, 2018.

[33] Christine Erbe, Sarah A. Marley, Renée P. Schoeman, Joshua N. Smith, Leah E. Trigg, and Clare Beth Embling. The effects of ship noise on marine mammals—a review. *Frontiers in Marine Science*, 6, oct 2019.

[34] Alison J. Fairbrass, Michael Firman, Carol Williams, Gabriel J. Brostow, Helena Titheridge, and Kate E. Jones. CityNet—Deep learning tools for urban ecoacoustic assessment. *Methods in Ecology and Evolution*, 10(2):186–197, nov 2018.

[35] Jialue Fan, Wei Xu, Ying Wu, and Yihong Gong. Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21(10):1610–1623, October 2010.

[36] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, August 2013.

[37] Laurene V. Fausett. *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Educations India, 1st edition, 1994.

[38] Maria Cristina Fossi, Matteo Baini, and Mark Peter Simmonds. Cetaceans as ocean health indicators of marine litter impact at global scale. *Frontiers in Environmental Science*, 8:586627, December 2020.

[39] Michael Franke and Judith Degen. The softmax function: Properties, motivation, and interpretation. September 2023.

[40] Fabio Frazao, Bruno Padovese, and Oliver S. Kirsebom. Workshop report: detection and classification in marine bioacoustics with deep learning. *arXiv:2002.08249v1*, 2020.

[41] Kazuki Fujimori, Bisser Raytchev, Kazufumi Kaneda, Yasufumi Yamada, Yu Teshima, Emyo Fujioka, Shizuko Hiryu, and Toru Tamaki. Localization of flying bats from multichannel audio signals by estimating location map with convolutional neural networks. *Journal of Robotics and Mechatronics*, 33(3):515–525, jun 2021.

[42] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980.

[43] Sean A. Fulop. Phonetics and signal processing. In *Speech Spectrum Analysis*, Signals and Communication Technology, pages 5–40. Springer Berlin Heidelberg, 2011.

[44] Sean A. Fulop. The Fourier power spectrum and spectrogram. In *Speech Spectrum Analysis*, Signals and Communication Technology, pages 69–106. Springer Berlin Heidelberg, 2011.

[45] Lola Gilbert, Tiphaine Jeanniard-du Dot, Matthieu Authier, Tiphaine Chouvelon, and Jérôme Spitz. Composition of cetacean communities worldwide shapes their contribution to ocean nutrient cycling. *Nature Communications*, 14:5823, September 2023.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive computation and machine learning series. MIT press, 2016.

[47] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[48] Elliott Gordon-Rodriguez, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P. Cunningham. Uses and abuses of the cross-entropy loss: case studies in modern deep learning. *arXiv:2011.05231v1*, November 2020.

[49] Ashish Kumar Gupta, Ayan Seal, Mukesh Prasad, and Pritee Khanna. Salient object detection techniques in computer vision—a survey. *Entropy*, 22(10):1174, October 2020.

[50] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2nd edition, 2009.

[51] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 2nd edition, July 1998.

[52] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE, 2017.

[53] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv:1712.00409v1*, 2017.

[54] Martin Hilbert and Priscila López. The world's technological capacity to store, communicate, and compute information. *Science*, 332(6025):60–65, apr 2011.

[55] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. *Deep Features for Text Spotting*, pages 512–528. Springer International Publishing, 2014.

[56] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer US, 2nd edition, 2021.

[57] Susan M. Jarvis, Nancy DiMarzio, Stephanie Watwood, Karin Dolan, and Ronald Morrissey. Automated detection and classification of beaked whale buzzes on bottom-mounted hydrophones. *Frontiers in Remote Sensing*, 3:941838, sep 2022.

[58] M. Johnson, P. T. Madsen, W. M. X. Zimmer, N. Aguilar de Soto, and P. L. Tyack. Foraging Blainville's beaked whales (*Mesoplodon densirostris*) produce distinct click types matched to different phases of echolocation. *Journal of Experimental Biology*, 209(24):5038–5050, dec 2006.

[59] M. P. Johnson and P. L. Tyack. A digital acoustic recording tag for measuring the response of wild marine mammals to sound. *IEEE Journal of Oceanic Engineering*, 28(1):3–12, jan 2003.

[60] Mark Johnson. Dtag-3. `https://soundtags.wp.st-andrews.ac.uk/dtags/dtag-3/`. Last viewed April 10, 2024.

[61] Mark Johnson, Peter T. Madsen, Walter M. X. Zimmer, Natacha Aguilar de Soto, and Peter L. Tyack. Beaked whales echolocate on prey. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 271:S383–S386, dec 2004.

[62] Alexis Joly, Hervé Goëau, Hervé Glotin, Concetto Spampinato, Pierre Bonnet, Willem-Pier Vellinga, Jean-Christophe Lombardo, Robert Planqué, Simone Palazzo, and Henning Müller. Biodiversity information retrieval through large scale content-based identification: a long-term evaluation. In *Information Retrieval Evaluation in a Changing World*, volume 41 of *The information retrieval series*, pages 389–413. Springer International Publishing, 2019.

[63] Benjamin A. Jones, Timothy K. Stanton, Andone C. Lavery, Mark P. Johnson, Peter T. Madsen, and Peter L. Tyack. Classification of broadband echoes from prey of a foraging Blainville's beaked whale. *The Journal of the Acoustical Society of America*, 123(3):1753–1762, mar 2008.

[64] Nasser Kehtarnavaz. Frequency domain processing. In *Digital Signal Processing System Design*, pages 175–196. Elsevier, 2nd edition, 2008.

[65] Robert D. Kenney and Howard E. Winn. Cetacean biomass densities near submarine canyons compared to adjacent shelf/slope areas. *Continental Shelf Research*, 7(2):107–114, 1987.

[66] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[67] Salman Khan, Hossein Rahmani, Syed Afaq Ali Shah, and Mohammed Bennamoun. *A Guide to Convolutional Neural Networks for Computer Vision.* Springer International Publishing, 2018.

[68] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[69] Ivan Kiskin, Davide Zilli, Yunpeng Li, Marianne Sinka, Kathy Willis, and Stephen Roberts. Bioacoustic detection with wavelet-conditioned convolutional neural networks. *Neural Computing and Applications*, 32(4):915–927, aug 2018.

[70] Daichi Kitaguchi, Nobuyoshi Takeshita, Hiro Hasegawa, and Masaaki Ito. Artificial intelligence–based computer vision in surgery: Recent advances and future perspectives. *Annals of Gastroenterological Surgery*, 6(1):29–36, October 2021.

[71] W. Koening. A new frequency scale for acoustic measurements. *Bell Laboratories Record*, 27:299–301, 1949.

[72] Sunil Kumar Kopparapu and M. Laxminarayana. Choice of Mel filter bank in computing MFCC of a resampled speech. In *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, pages 121–124. IEEE, may 2010.

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2012.

[74] Russell Lande and Susan Shannon. The role of genetic variation in adaptation and population persistence in a changing environment. *Evolution*, 50(1):434, February 1996.

[75] Jack LeBien, Ming Zhong, Marconi Campos-Cerqueira, Julian P. Velev, Rahul Dodhia, Juan Lavista Ferres, and T. Mitchell Aide. A pipeline for identification of bird and frog species in tropical soundscape recordings using a convolutional neural network. *Ecological Informatics*, 59:101113, sep 2020.

[76] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015.

[77] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[78] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, apr 2020.

[79] Murray J. Littlejohn. Historical aspects of recording and analysis in anuran bioacoustics: 1954–1997. *Bioacoustics*, 9(1):69–80, jan 1998.

[80] Richard G. Lyons. *Understanding digital signal processing.* Prentice Hall, 3rd edition, 2011.

[81] C. Macleod and G. Mitchell. Known key areas for beaked whales around the world. *Cetacean Research and Management*, 7(3):309–322, 2006.

[82] Colin Macleod, Nan Hauser, and Hoyt Op. Diversity, relative density and structure of the cetacean community in summer months east of Great Abaco, Bahamas. *Journal of the Marine Biological Association of the United Kingdom*, 84:469–474, 2004.

[83] Colin D. MacLeod and Alain F. Zuur. Habitat utilization by Blainville's beaked whales off Great Abaco, northern Bahamas, in relation to seabed topography. *Marine Biology*, 147(1):1–11, jan 2005.

[84] Shyam Madhusudhana, Gianni Pavan, Lee A. Miller, William L. Gannon, Anthony Hawkins, Christine Erbe, Jennifer A. Hamel, and Jeanette A. Thomas. Choosing equipment for animal bioacoustic research. In *Exploring Animal Behavior Through Sound*, volume 1, pages 37–85. Springer International Publishing, 2022.

[85] P. T. Madsen, M. Johnson, N. Aguilar de Soto, W. M. X. Zimmer, and P. Tyack. Biosonar performance of foraging beaked whales (*Mesoplodon densirostris*). *Journal of Experimental Biology*, 208(2):181–194, jan 2005.

[86] Paul Magron, Roland Badeau, and Bertrand David. Phase reconstruction of spectrograms with linear unwrapping: Application to audio signal restoration. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, aug 2015.

[87] Farhad Maleki, Nikesh Muthukrishnan, Katie Ovens, Caroline Reinhold, and Reza Forghani. Machine learning algorithm validation. *Neuroimaging Clinics of North America*, 30(4):433–445, November 2020.

[88] T. A. Marques, L. Munger, L. Thomas, S. Wiggins, and J. A. Hildebrand. Estimating north pacific right whale *Eubalaena japonica* density using passive acoustic cue counting. *Endangered Species Research*, 13(3):163–172, March 2011.

[89] Tiago A. Marques, Len Thomas, Stephen W. Martin, David K. Mellinger, Jessica A. Ward, David J. Moretti, Danielle Harris, and Peter L. Tyack. Estimating animal population density using passive acoustics. *Biological Reviews*, 88(2):287–309, nov 2012.

[90] Tiago A. Marques, Len Thomas, Jessica Ward, Nancy DiMarzio, and Peter L. Tyack. Estimating cetacean population density using fixed passive acoustic sensors: An example with Blainville's beaked whales. *The Journal of the Acoustical Society of America*, 125(4):1982–1994, apr 2009.

[91] Iain J. Marshall and Byron C. Wallace. Toward systematic review automation: a practical guide to using machine learning tools in research synthesis. *Systematic Reviews*, 8(1), July 2019.

[92] R. McEachern. How the ear really works. In *[1992] Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pages 437–440. IEEE.

[93] Brian McFee, McVicar Matt, Faronbi Daniel, Roman Iran, Gover Matan, Balke Stefan, Seyfarth Scott, Malek Ayoub, Raffel Colin, Lostanlen Vincent, van Niekirk Benjamin, Lee Dana, Cwitkowitz Frank, Zalkow Frank, Nieto Oriol, Ellis Dan, Mason Jack, Lee Kyungyun, Steers Bea, Halvachs Emily, Thomé Carl, Robert-Stöter Fabian, Bittner Rachel, Wei Ziyao, Weiss Adam, Battenberg Eric, Choi Keunwoo, Yamamoto Ryuichi, Carr C., J., Metsai Alex, Sullivan Stefan, Friesch Pius, Krishnakumar Asmitha, Hidaka Shunsuke, Kowalik Steve, Keller Fabian, Mazur Dan, Chabot-Leclerc Alexandre, Hawthorne Curtis, Ramaprasad Chandrashekhar, Keum Myungchul, Gomez Juanita, Monroe Will, Morozov Viktor, Andreevitch, Eliasi Kian, nullmightybofo, Biberstein Paul, Sergin N., Dorukhan, Hennequin Romain, Naktinis Rimvydas, beantowel, Kim Taewoon, Åsen Jon, Petter, Lim Joon, Malins Alex, Hereñú Darío, van der Struijk Stef, Nickel Lorenz, Wu Jackie, Wang Zhen, Gates Tim, Vollrath Matt, Sarroff Andy, Xiao-Ming, Porter Alastair, Kranzler Seth, Voodoohop, Di Gangi Mattia, Jinoz Helmi, Guerrero Connor, Mazhar Abduttayyeb, toddrme2178, Baratz Zvi, Kostin Anton, Zhuang Xinlu, Lo Cash, TingHin, Campr Pavel, Semeniuc Eric, Biswal Monsij, Moura Shayenne, Brossier Paul, Lee Hojin, and Pimenta Waldir. librosa/librosa: 0.10.1. `https://doi.org/10.5281/zenodo.8252662`, 2023. Last assessed 20 March 2024.

[94] Xinhao Mei, Xubo Liu, Qiushi Huang, Mark D. Plumbley, and Wenwu Wang. Audio captioning transformer. *arXiv:2107.09817v1*, July 2021.

[95] Marvin Lee Minsky and Seymour Papert. *Perceptrons An Introduction to Computational Geometry*. Brand: MIT Press, 1969.

[96] Tom M. Mitchell. Does machine learning really work? *AI Magazine*, 18(3):11, Sep. 1997.

[97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[98] Christoph Molnar. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Independently published, 2020.

[99] Berndt Müller, Joachim Reinhardt, and Michael T. Strickland. *Neural Networks: An Introduction*. Physics of Neural Networks. Springer Science & Business Media, 2nd edition, 1995.

[100] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[101] D. Kanishka Nithin and P. Bagavathi Sivakumar. Generic feature learning in computer vision. *Procedia Computer Science*, 58:202–209, 2015.

[102] A. Noumida and Rajeev Rajan. Deep learning-based automatic bird species identification from isolated recordings. In *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, pages 252–256. IEEE, 2021.

[103] Chigozie Enyinna Nwankpa. Advances in optimisation algorithms and techniques for deep learning. *Advances in Science, Technology and Engineering Systems Journal*, 5(5):563–577, 2020.

[104] Alan V. Oppenheim. Speech spectrograms using the fast Fourier transform. *IEEE Spectrum*, 7(8):57–62, aug 1970.

[105] Douglas O'Shaughnessy. *Speech Communications: Human and Machine*. Wiley-IEEE Press, 2nd edition, 1999.

[106] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[107] Andrew J. Oxenham. How We Hear: The Perception and Neural Coding of Sound. *Annual Review of Psychology*, 69(1):27–50, jan 2018.

[108] Hanna Pamula, Agnieszka Pocha, and Maciej Klaczynski. Towards the acoustic monitoring of birds migrating at night. *Biodiversity Information Science and Standards*, 3:e36589, jun 2019.

[109] Olivier C. Pasche and Sebastian Engelke. Neural networks for extreme quantile regression with an application to forecasting of flood risk. *arXiv:2208.07590v3*, 2022.

[110] Justin S. Paul, Andrew J. Plassard, Bennett A. Landman, and Daniel Fabbri. Deep learning for brain tumor classification. In Andrzej Krol and Barjor Gimi, editors, *SPIE Proceedings*, volume 10137. SPIE, March 2017.

[111] Branco Paula, Luis Torgo, and Rita Ribeiro. A survey of predictive modelling under imbalanced distributions. *arXiv:1505.01658v2*, 2015.

[112] Gianni Pavan. Short field course on bioacoustics. *Taxonomy Summer School*, pages 1–15, 2008.

[113] Gianni Pavan, Gregory Budney, Holger Klinck, Hervé Glotin, Dena J. Clink, and Jeanette A. Thomas. History of sound recording and analysis equipment. In *Exploring Animal Behavior Through Sound: Volume 1*, chapter 1, pages 1–36. Springer International Publishing, 2022.

[114] Bryan C. Pijanowski, Luis J. Villanueva-Rivera, Sarah L. Dumyahn, Almo Farina, Bernie L. Krause, Brian M. Napoletano, Stuart H. Gage, and Nadia Pieretti. Soundscape ecology: The science of sound in the landscape. *BioScience*, 61(3):203–216, March 2011.

[115] Robert Pitman. *Encyclopedia of Marine Mammals*. Elsevier Inc., 3rd edition, 2018.

[116] Ken C. Pohlmann. *Principles of digital audio*. McGraw-Hill Professional, 6th edition, 2000.

[117] S.J.D. Prince. *Computer Vision: Models Learning and Inference*. Cambridge University Press, 1st edition, 2012.

[118] Richard Ranft. Natural sound archives: past, present and future. *Anais da Academia Brasileira de Ciências*, 76(2):456–460, jun 2004.

[119] Shannon Rankin, Taiki Sakai, Frederick I. Archer, Jay Barlow, Danielle Cholewiak, Annamaria I. DeAngelis, Jennifer L.K. McCullough, Erin M. Oleson, Anne E. Simonis, Melissa S. Soldevilla, and Jennifer S. Trickey. Open-source machine learning banter acoustic classification of beaked whale echolocation pulses. *Ecological Informatics*, 80:102511, May 2024.

[120] C. R. Rao and Venkat N. Gudivada. *Computational analysis and understanding of natural languages : principles, methods and applications, Volume 38*. Elsevier Science Technology Books, 2018.

[121] J. E. Reynolds, W. F. Perrin, R. R. Reeves, S. Moore, and T. J. Regan. *Marine Mammal Research*. The Johns Hopkins University Press, 2005.

[122] Hugh Robjohns. A brief history of microphones. `https://micpedia.com/brief-history-of-microphones/`, 2001.

[123] Paul Roe, Philip Eichinski, Richard A. Fuller, Paul G. McDonald, Lin Schwarzkopf, Michael Towsey, Anthony Truskinger, David Tucker, and David M. Watson. The australian acoustic observatory. *Methods in Ecology and Evolution*, 12(10):1802–1808, jul 2021.

[124] Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, May 2004.

[125] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[126] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 1 2016.

[127] Stuart J. Russell. *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., 3rd edition, 2010.

[128] Muhammad F. Safdar, Piotr Pałka, Ahmed Al Faresi, and Robert M. Nowak. Optimizing electrocardiogram signal augmentation for realistic synthetic data in deep learning model. In *2024 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 54–59. IEEE, sep 2024.

[129] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[130] Elena Schall, Idil Ilgaz Kaya, Elisabeth Debusschere, Paul Devos, and Clea Parcerisas. Deep learning in marine bioacoustics: a benchmark for baleen whale detection. *Remote Sensing in Ecology and Conservation*, April 2024.

[131] Marianne E. Sinka, Davide Zilli, Yunpeng Li, Ivan Kiskin, Dickson Msaky, Japhet Kihonda, Gustav Mkandawile, Emmanuel Kaindoa, Gerry Killeen, Josue Zanga, Emery Metelo, Daniel Kirkham, Paul Mansiangi, Eva Herreros-Moya, Waqas Rafique, Theeraphap Chareonviriyaphap, Rungarun Tisgratog, Lawrence Wang, Henry Chan, Benjamin Gutteridge, Henry Portwood, Stephen Roberts, and Kathy J. Willis. HumBug – an acoustic mosquito monitoring tool for use on budget smartphones. *Methods in Ecology and Evolution*, 12(10):1848–1859, jul 2021.

[132] Elias Sprengel, Martin Jaggi, Yannic Kilcher, and Thomas Hofmann. Audio based bird species identification using deep learning techniques. In *Conference and Labs of the Evaluation Forum (CLEF) 2016*, pages 547–559. LifeCLEF 2016, September 2016.

[133] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[134] S. S. Stevens and J. Volkmann. The relation of pitch to frequency: A revised scale. *The American Journal of Psychology*, 53(3):329, jul 1940.

[135] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, jan 1937.

[136] Dan Stowell. Computational bioacoustics with deep learning: a review and roadmap. *PeerJ*, 10:e13152, March 2022.

[137] Dan Stowell, Tereza Petrusková, Martin Šálek, and Pavel Linhart. Automatic acoustic identification of individuals in multiple species: improving identification across recording conditions. *Journal of The Royal Society Interface*, 16(153):20180940, apr 2019.

[138] Richard Sutton. Two problems with back propagation and other steepest descent learning procedures for networks. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986*, pages 823–832, 1986.

[139] Michael A. Tabak, Kevin L. Murray, Ashley M. Reed, John A. Lombardi, and Kimberly J. Bay. Automated classification of bat echolocation call recordings with artificial intelligence. *Ecological Informatics*, 68:101526, 2022.

[140] Len Thomas and Tiago A. Marques. Passive acoustic monitoring for estimating animal density. *Acoustics Today*, 8(3):35–44, 2012.

[141] Todor Ganchev. *Computational Bioacoustics: Biodiversity Monitoring and Assessment*, volume 4 of *Speech technology and text mining in medicine and healthcare*. Walter de Gruyter Incorporated, 2017, jun 2017.

[142] Alexander Toshev and Christian Szegedy. DeepPose: human pose estimation via deep neural networks. In *[2014] IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660. IEEE, June 2014.

[143] Michael Towsey, Birgit Planitz, Alfredo Nantes, Jason Wimmer, and Paul Roe. A toolbox for animal call recognition. *Bioacoustics*, 21(2):107–125, jun 2012.

[144] Peter L. Tyack, Mark Johnson, Natacha Aguilar Soto, Albert Sturlese, and Peter T. Madsen. Extreme diving of beaked whales. *Journal of Experimental Biology*, 209(21):4238–4253, nov 2006.

[145] Peter L. Tyack, Mark P. Johnson, Walter M. X. Zimmer, Natacha Aguilar De Soto, and Peter T. Madsen. Acoustic behavior of beaked whales, with implications for acoustic monitoring. In *OCEANS 2006*, pages 1–6. IEEE, 2006.

[146] Peter L. Tyack, Walter M. X. Zimmer, David Moretti, Brandon L. Southall, Diane E. Claridge, John W. Durban, Christopher W. Clark, Angela D'Amico, Nancy DiMarzio, Susan Jarvis, Elena McCarthy, Ronald Morrissey, Jessica Ward, and Ian L. Boyd. Beaked whales respond to simulated and actual navy sonar. *PLoS ONE*, 6(3):e17009, March 2011.

[147] S. Umesh, L. Cohen, and D. Nelson. Fitting the mel scale. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, pages 217–220. IEEE, 1999.

[148] Samir B. Unadkat, Mãlina M. Ciocoiu, and Larry R. Medsker. *Recurrent neural networks: design and applications*, chapter 1. The CRC Press International Series on Computational Intelligence. CRC press, 1999.

[149] Robert J. Urick. *Principles of underwater sound*. McGraw-Hill, 2nd edition, 1975.

[150] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 30:6000–6010, December 2017.

[151] Fleur Visser, Machiel G. Oudejans, Onno A. Keller, Peter T. Madsen, and Mark Johnson. Sowerby's beaked whale biosonar and movement strategy indicate deep-sea foraging niche differentiation in mesoplodont whales. *Journal of Experimental Biology*, 225(9):jeb243728, may 2022.

[152] David H. von Seggern. *CRC Standard Curves and Surfaces with Mathematica*. Chapman Hall/CRC applied Mathematics and Nonlinear Science Series. Chapman Hall/CRC, 2nd edition, 2017.

[153] Emily E. Waddell, Jeppe H. Rasmussen, and Ana Širović. Applying artificial intelligence methods to detect and classify fish calls from the northern gulf of Mexico. *Journal of Marine Science and Engineering*, 9(10):1128, oct 2021.

[154] Changhong Wang, Emmanouil Benetos, Shuge Wang, and Elisabetta Versace. Joint scattering for automatic chick call recognition. In *[2022] 30th European Signal Processing Conference (EUSIPCO)*, pages 195–199. IEEE, August 2022.

[155] Qi Wang. *A comprehensive survey of loss functions in machine learning*, volume 9 of *Annals of Data Science*, pages 187–212. Springer, 2022.

[156] G. Waring, T. Hamazaki, Daniel Sheehan, and Grayson Wood. Characterization of beaked whale (*Ziphiidae*) and sperm whale (*Physeter macrocephalus*) summer habitat in shelf-edge and deeper waters off the northeast U.S. *Marine Mammal Science*, 17(4):703–717, 2001.

[157] Ben G. Weinstein. A computer vision for animal ecology. *Journal of Animal Ecology*, 87(3):533–545, November 2017.

[158] I. H. Witten and Eibe. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, 2nd edition, 2005.

[159] Rui Zhao, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Saliency detection by multi-context deep learning. In *[2015] IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1265–1274. IEEE, June 2015.

[160] Ming Zhong, Manuel Castellote, Rahul Dodhia, Juan Lavista Ferres, Mandy Keogh, and Arial Brewer. Beluga whale acoustic signal classification using deep learning neural network models. *The Journal of the Acoustical Society of America*, 147(3):1834–1841, March 2020.

[161] Morgan A. Ziegenhorn, Kaitlin E. Frasier, John A. Hildebrand, Erin M. Oleson, Robin W. Baird, Sean M. Wiggins, and Simone Baumann-Pickering. Discriminating and classifying odontocete echolocation clicks in the hawaiian islands using machine learning methods. *PLOS ONE*, 17(4):e0266424, April 2022.

Appendix

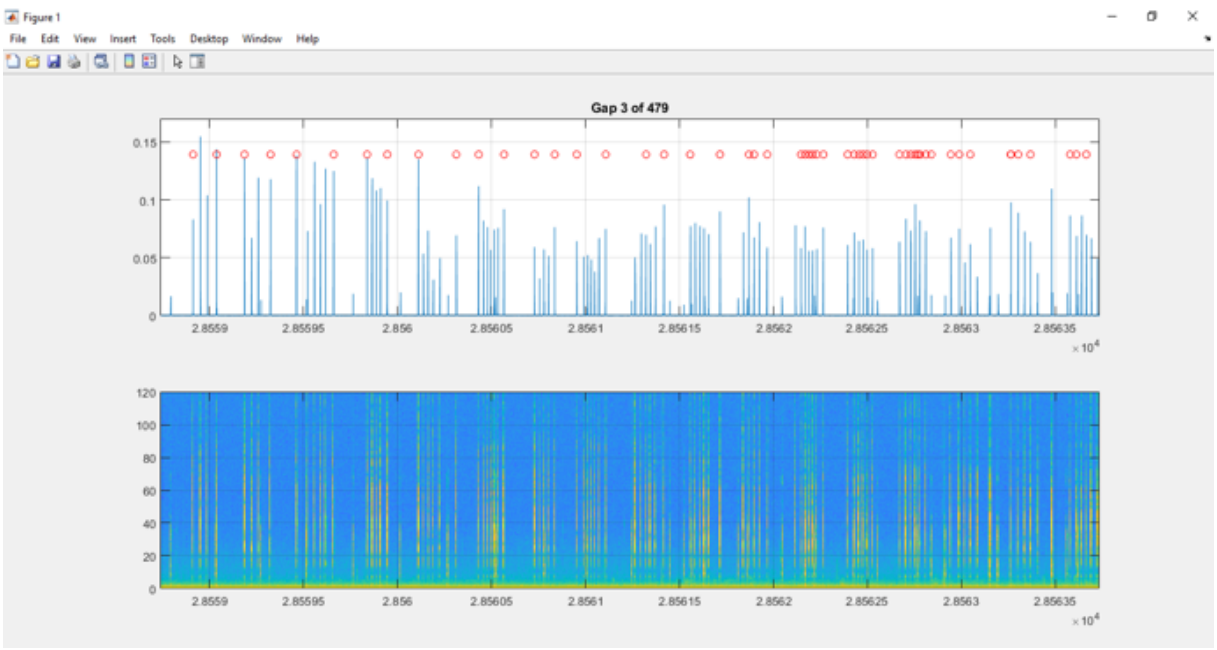


Figure 1: Image of the display shown by the Matlab tool used during the annotation process. The red dots correspond to the clicks detected. In this scenario there are clicks misidentified, requiring the user to add or remove red dots accordingly with the presence or absence of a click in that time instance.

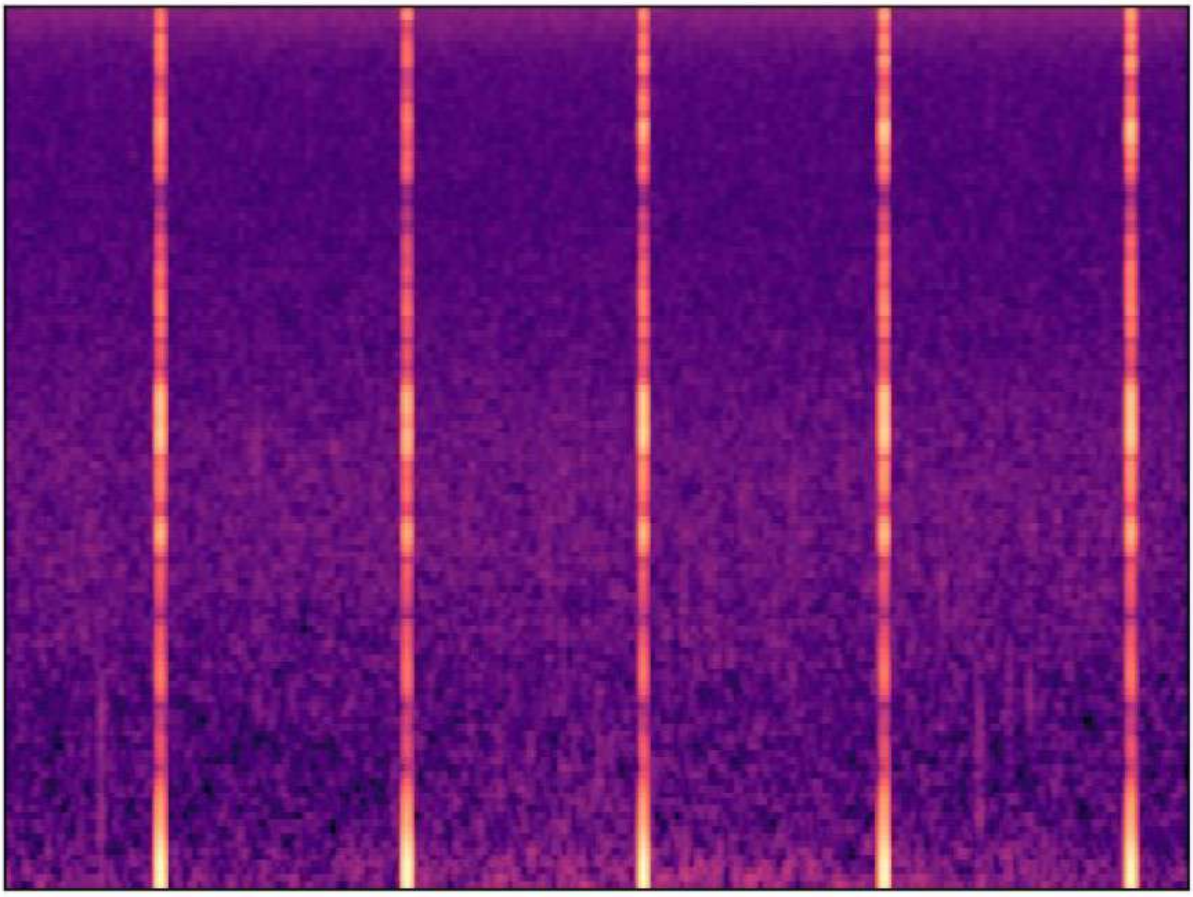


Figure 2: Spectrogram of Blainville's beaked whale recording, respective of Scenario 1 where only spectrograms of Blainville's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the presence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 94% that this is a spectrogram with Blainville's clicks and around 6% that this is a spectrogram without clicks. This means this case corresponds to a True Positive.

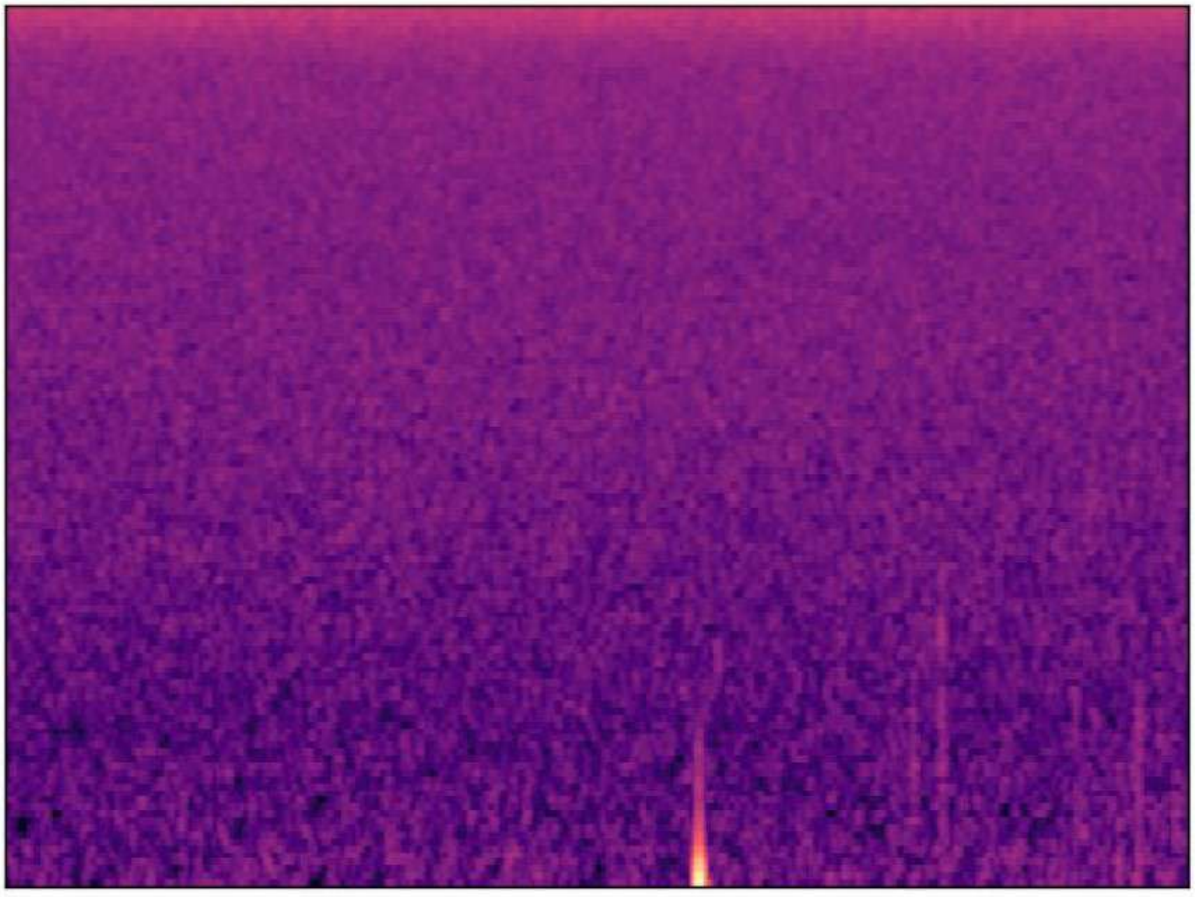


Figure 3: Spectrogram of Blainville's beaked whale recording, respective of Scenario 1 where only spectrograms of Blainville's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the absence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 99% that this is a spectrogram without Blainville's clicks and around 1% that this is a spectrogram with clicks. This means this case corresponds to a True Negative.

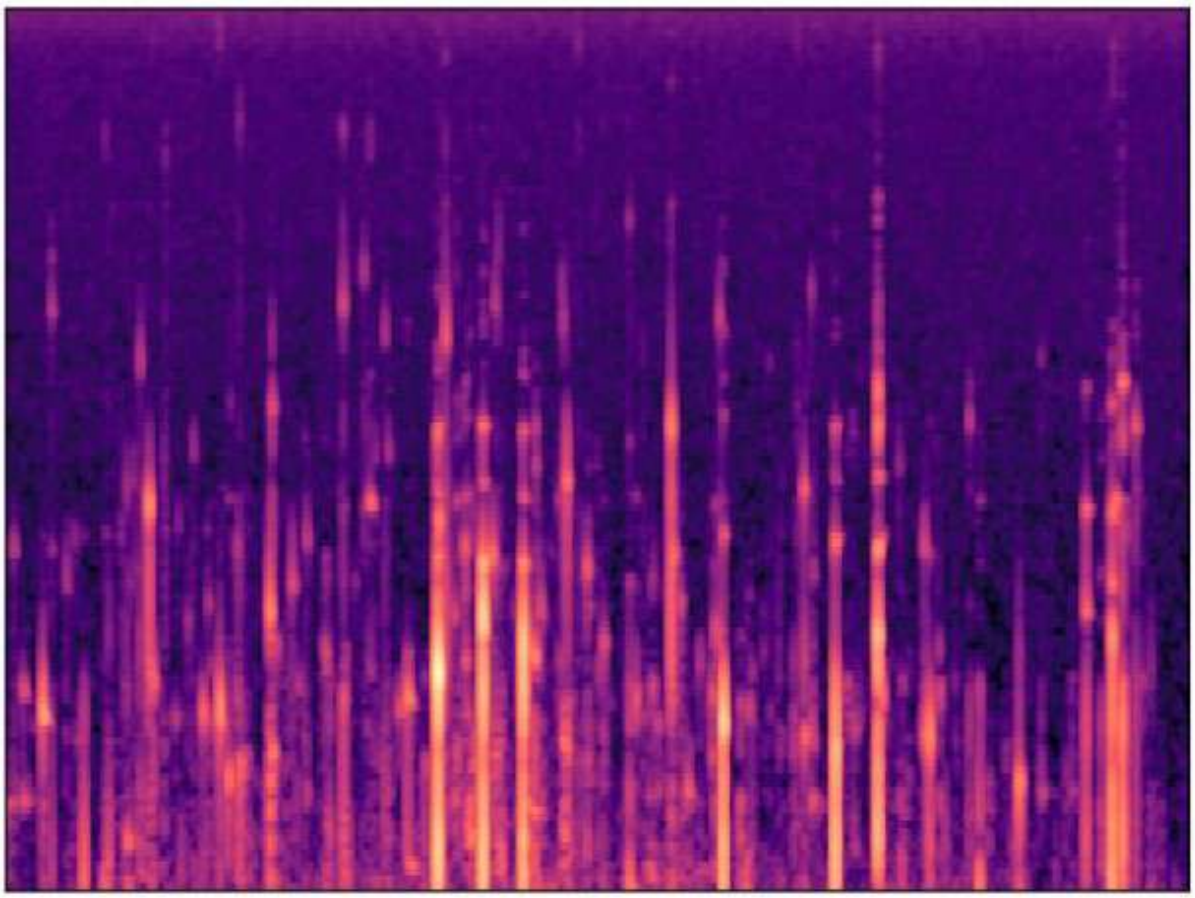


Figure 4: Spectrogram of Blainville's beaked whale recording, respective of Scenario 1 where only spectrograms of Blainville's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the absence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 43% that this is a spectrogram without Blainville's clicks and around 57% that this is a spectrogram with clicks. This means this case corresponds to a False Positive.

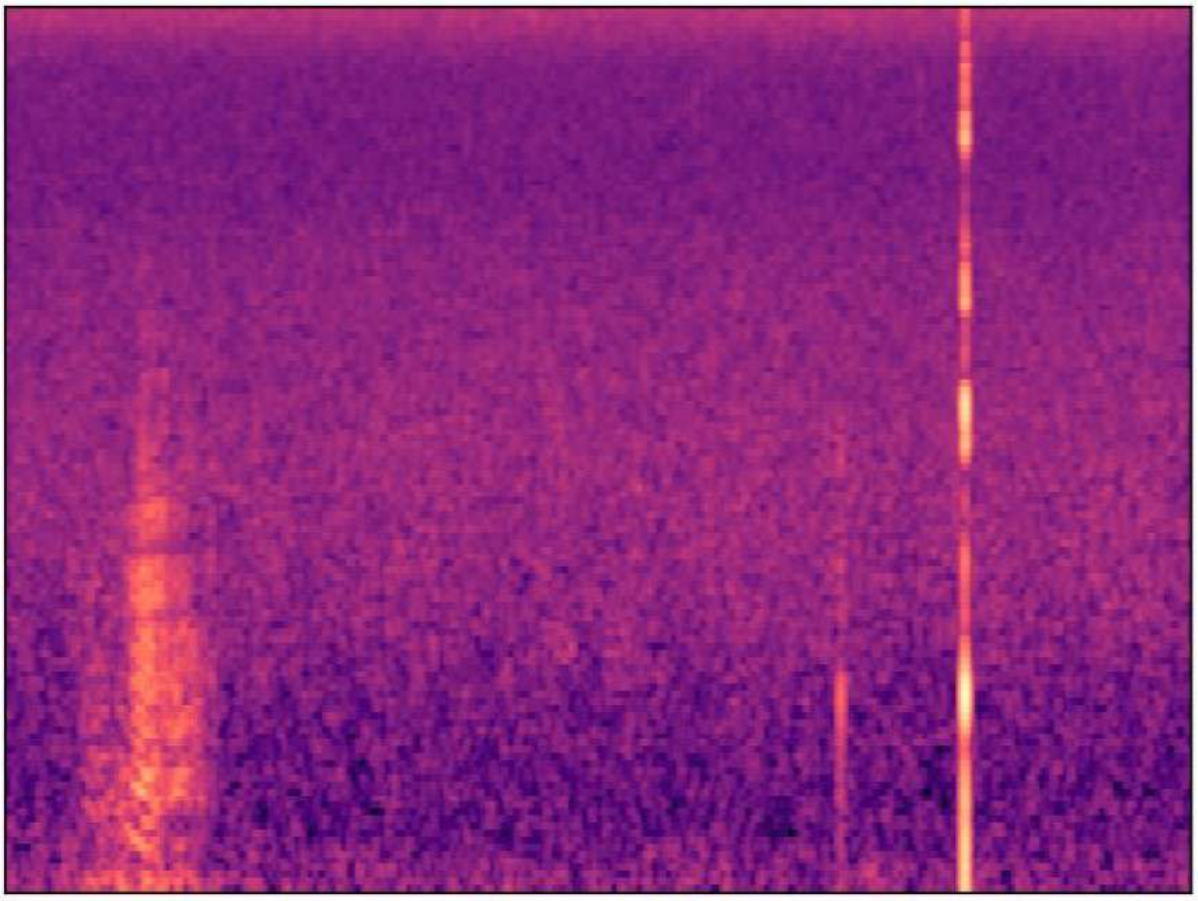


Figure 5: Spectrogram of Blainville's beaked whale recording, respective of Scenario 1 where only spectrograms of Blainville's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the presence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 72% that this is a spectrogram without Blainville's clicks and around 28% that this is a spectrogram with clicks. This means this case corresponds to a False Negative.

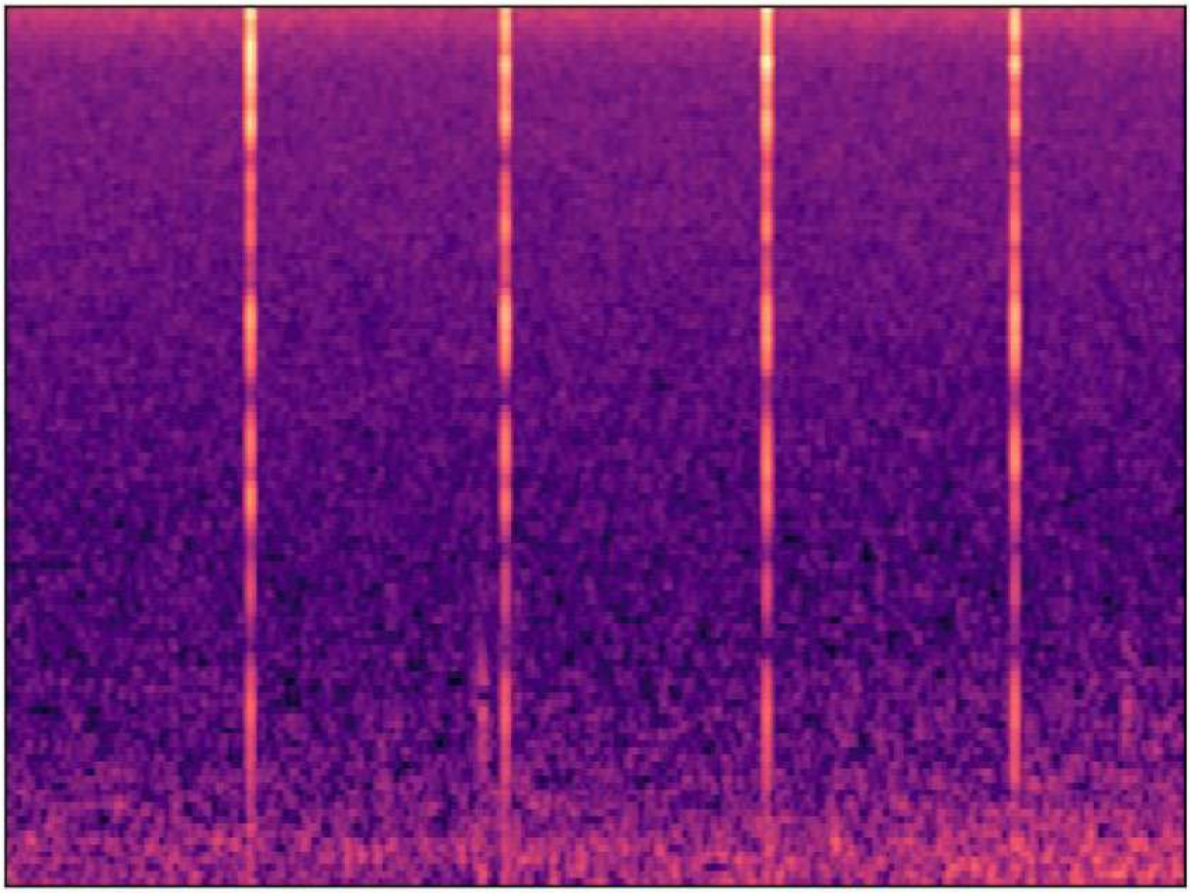


Figure 6: Spectrogram of Cuvier's beaked whale recording, respective of Scenario 1 where only spectrograms of Cuvier's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the presence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 89% that this is a spectrogram with Cuvier's clicks and around 11% that this is a spectrogram without clicks. This means this case corresponds to a True Positive.

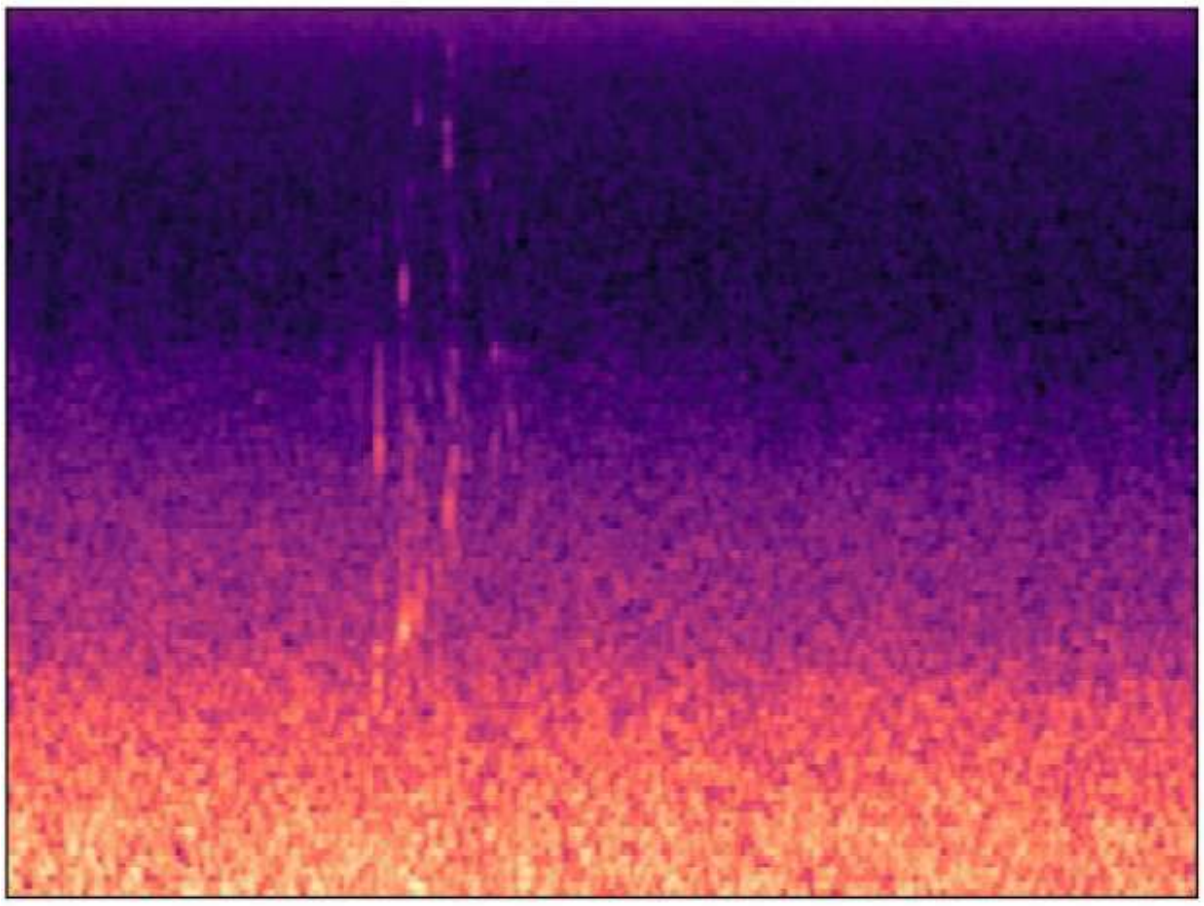


Figure 7: Spectrogram of Cuvier's beaked whale recording, respective of Scenario 1 where only spectrograms of Cuvier's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the absence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 99.55% that this is a spectrogram without Cuvier's clicks and around 0.45% that this is a spectrogram with clicks. This means this case corresponds to a True Negative.

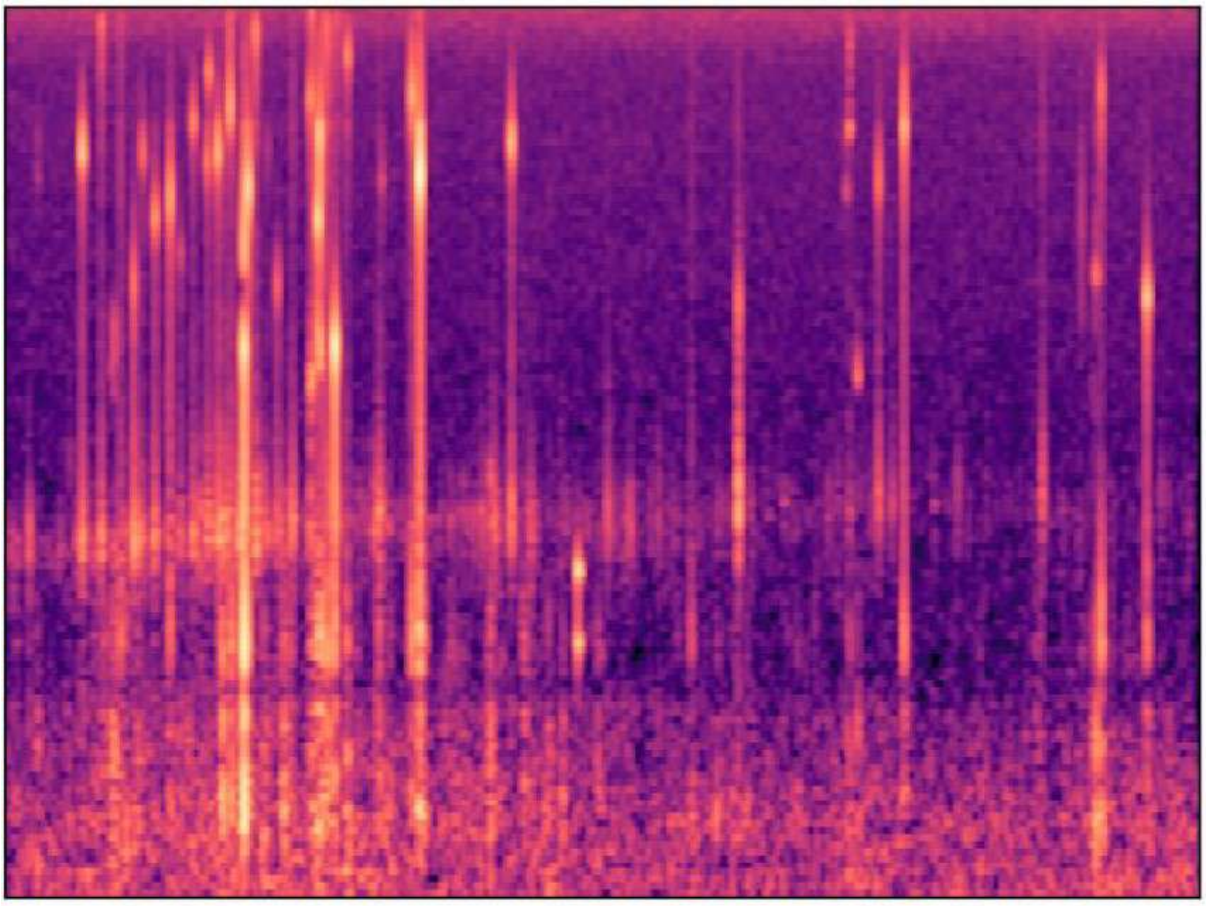


Figure 8: Spectrogram of Cuvier's beaked whale recording, respective of Scenario 1 where only spectrograms of Cuvier's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the absence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 36% that this is a spectrogram without Cuvier's clicks and around 64% that this is a spectrogram with clicks. This means this case corresponds to a False Positive.

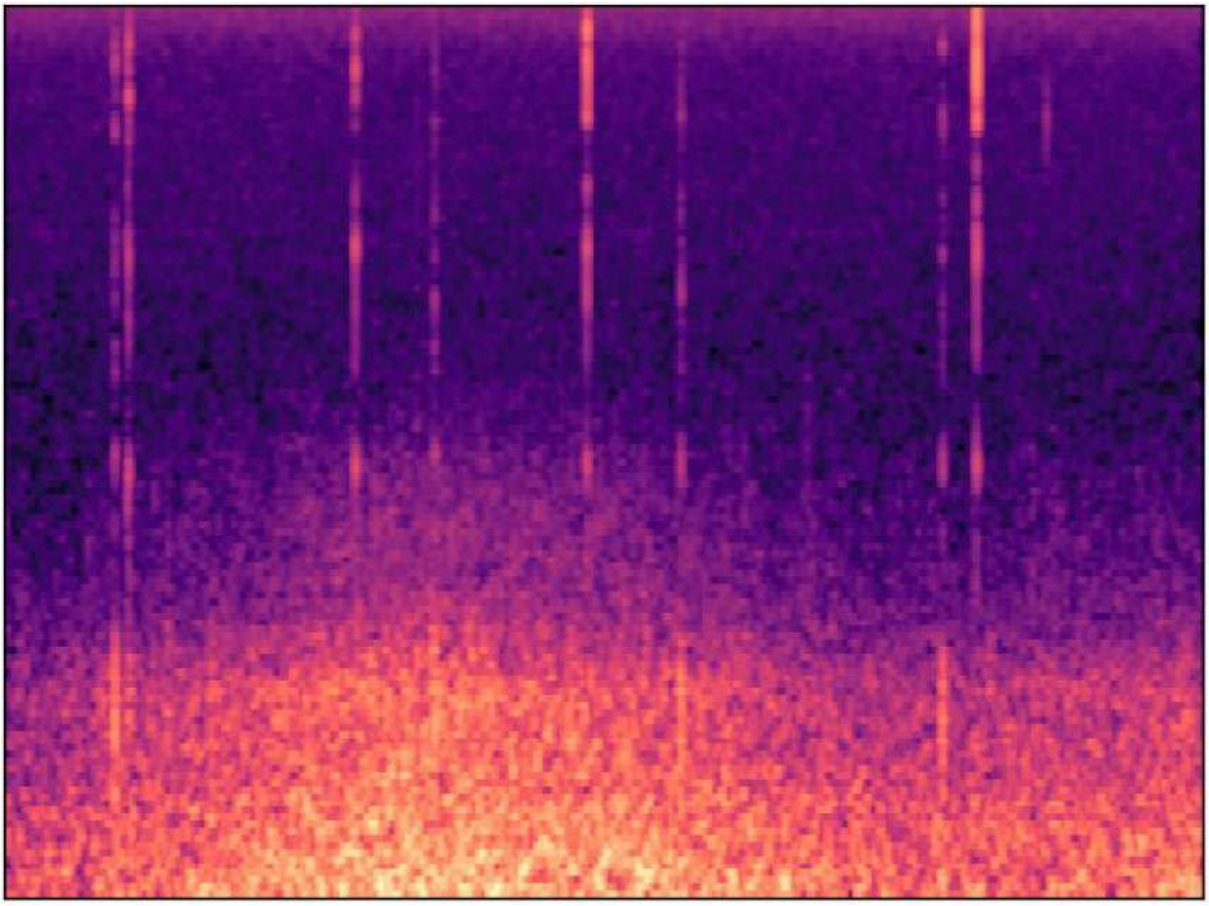


Figure 9: Spectrogram of Cuvier's beaked whale recording, respective of Scenario 1 where only spectrograms of Cuvier's clicks and spectrograms without clicks are present. The spectrogram true label corresponds to the presence of clicks, and the prediction made by the model is shown through the values below. In this case the model showed a certainty of approximately 98% that this is a spectrogram without Cuvier's clicks and around 2% that this is a spectrogram with clicks. This means this case corresponds to a False Negative.

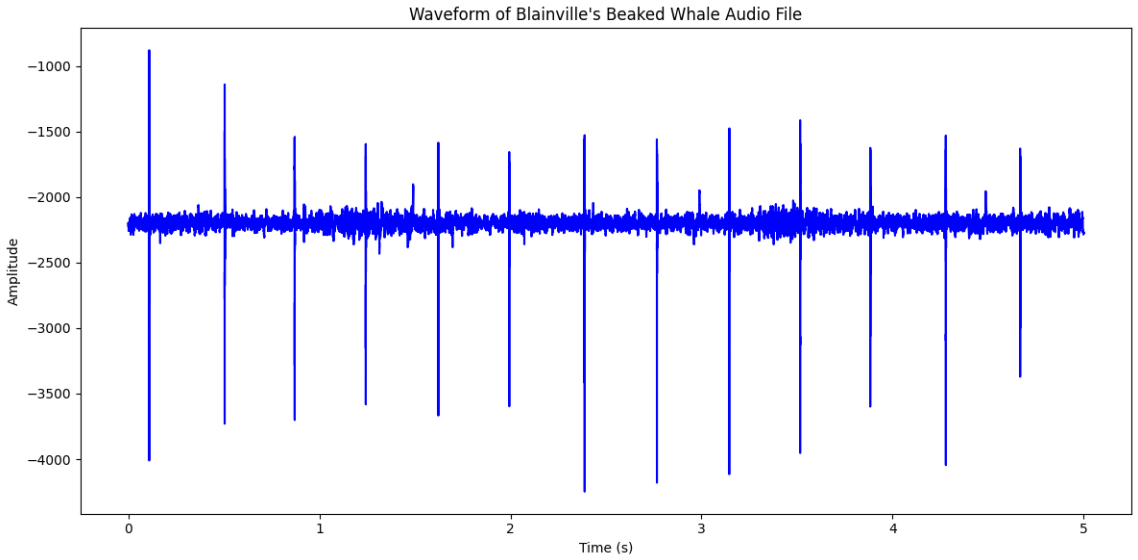


Figure 10: Representation of a 5 second segment of the waveform of Blainville's beaked whale clicks.

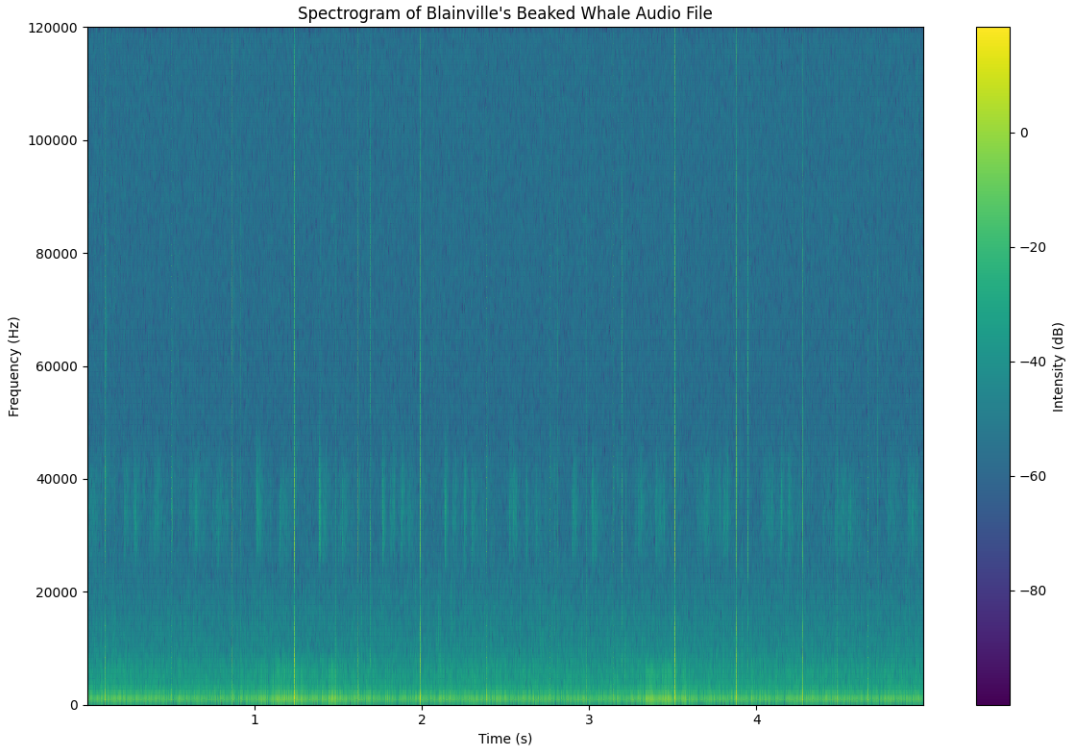


Figure 11: Representation of the transformation of Figure 10 into a mel spectrogram.

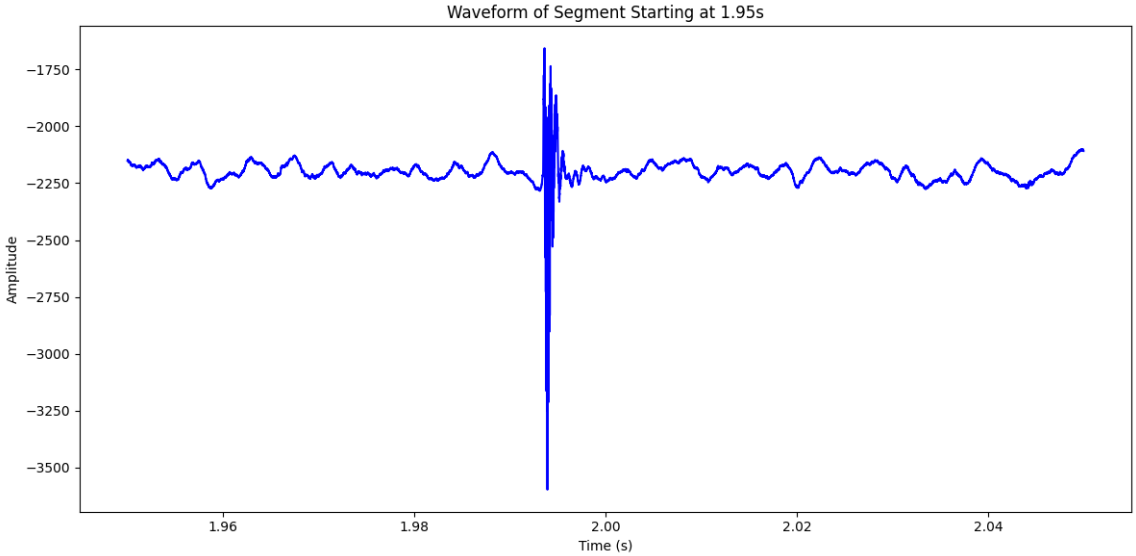


Figure 12: Representation of a portion of 0.5 seconds of the 5 second segment of the waveform of Blainville's beaked whale clicks presented in Figure. 10.

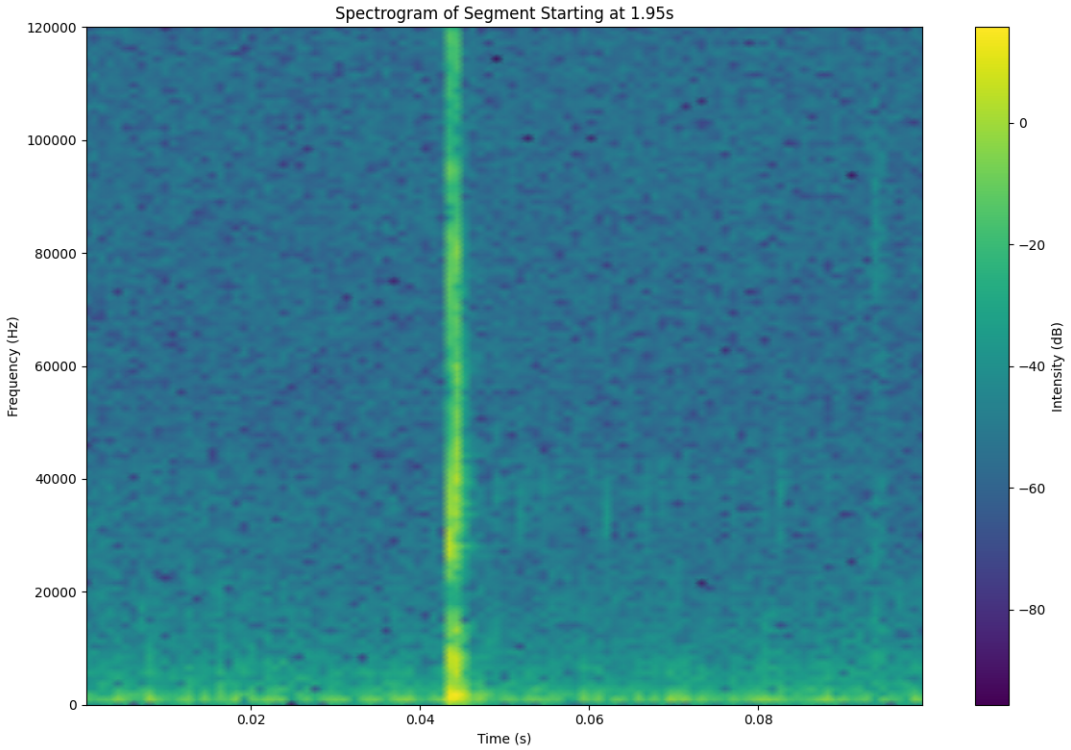


Figure 13: Representation of a portion of the 5 second segment of the mel spectrogram of Blainville's beaked whale clicks presented in Figure. 11, corresponding to the same time interval of the waveform shown in Figure. 12.

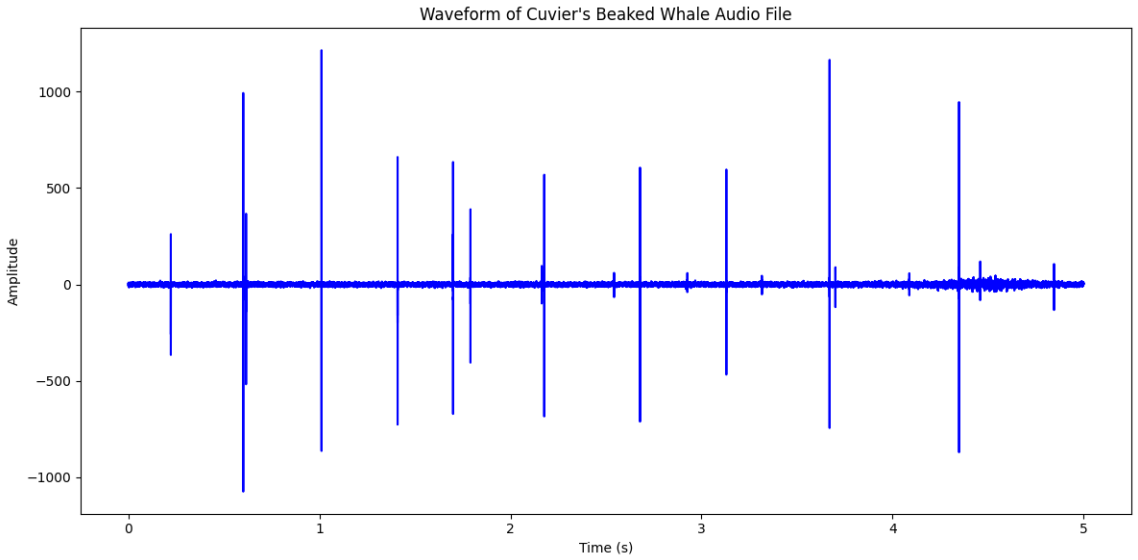


Figure 14: Representation of a 5 second segment of the waveform of Cuvier's beaked whale clicks.

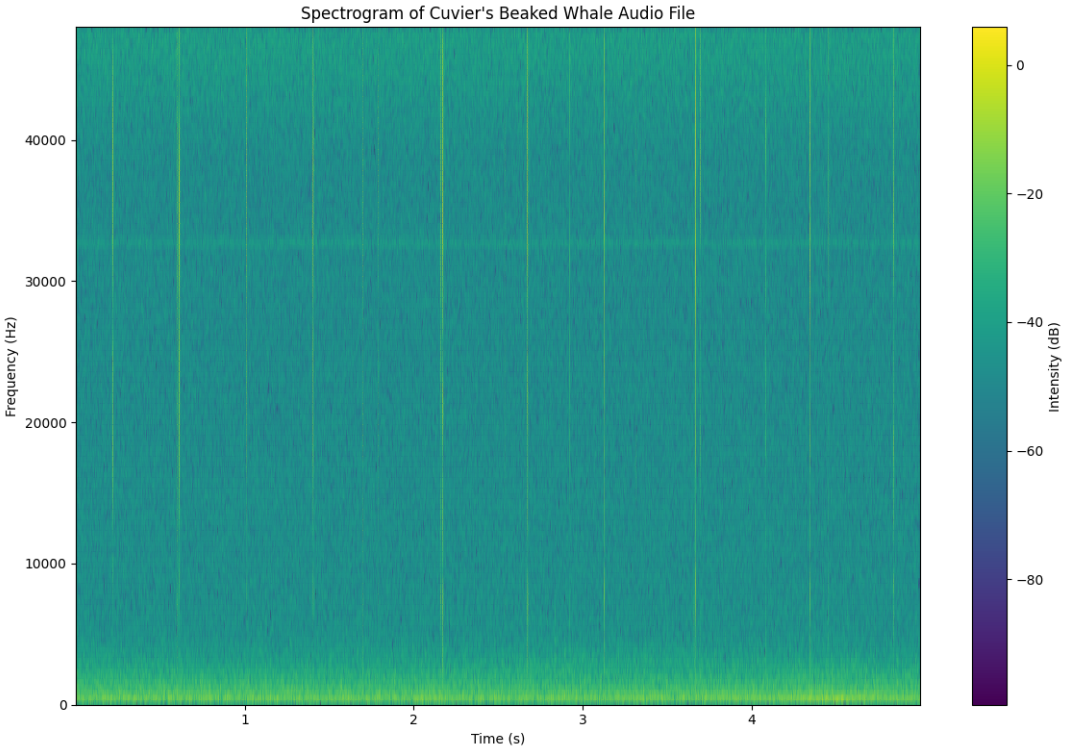


Figure 15: Representation of the transformation of Figure 14 into a mel spectrogram.

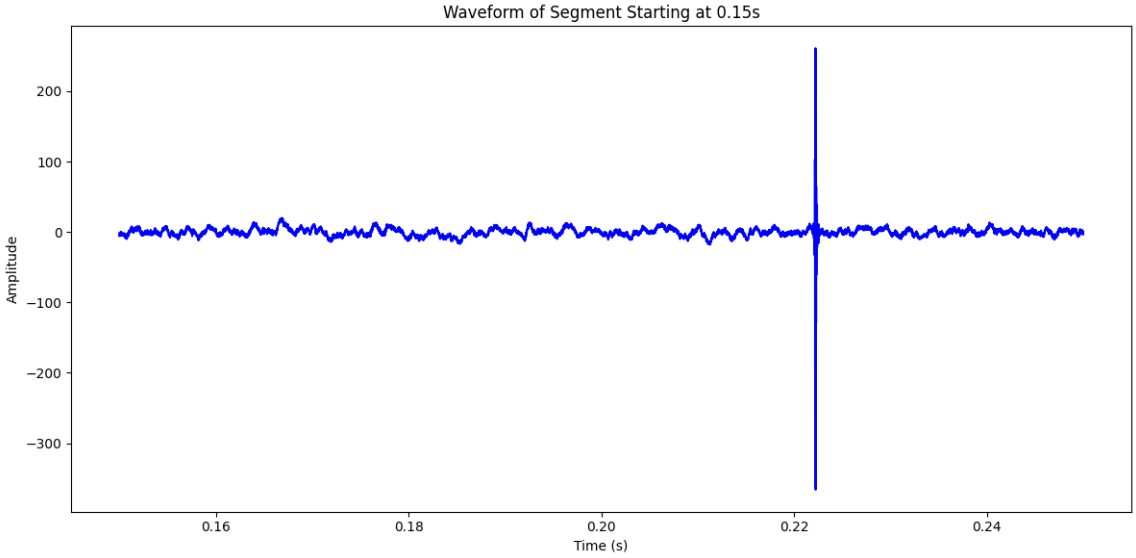


Figure 16: Representation of a portion of 0.5 seconds of the 5 second segment of the waveform of Cuvier's beaked whale clicks presented in Figure. 14.

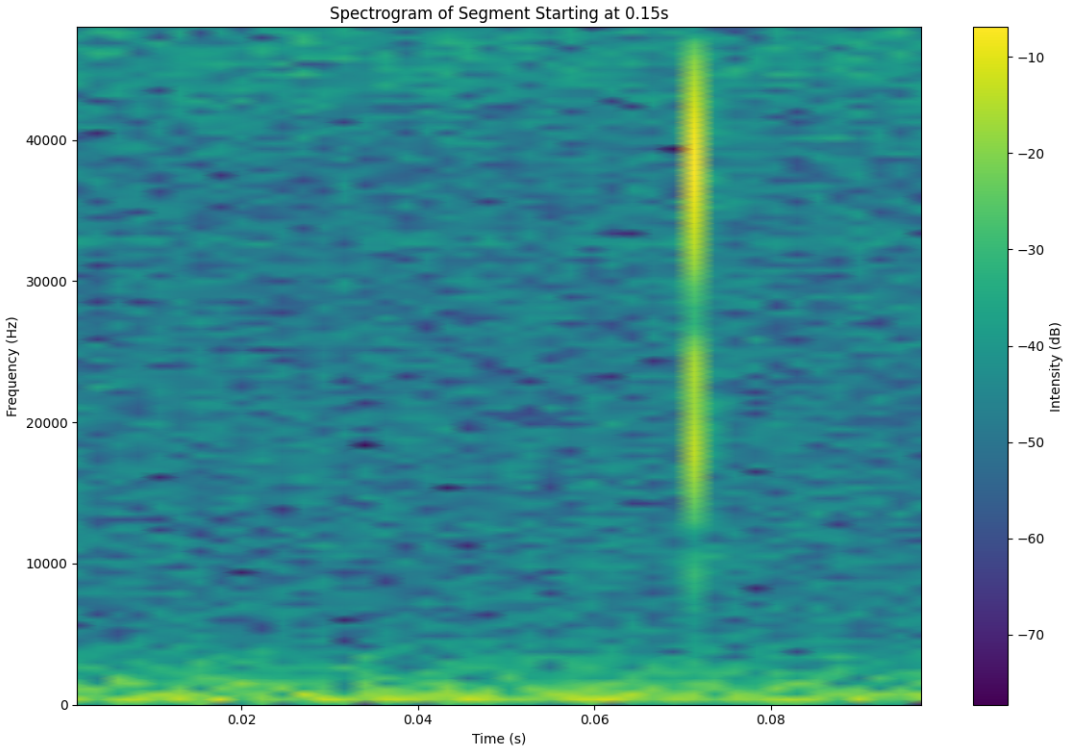


Figure 17: Representation of a portion of the 5 second segment of the mel spectrogram of Cuvier's beaked whale clicks presented in Figure. 15, corresponding to the same time interval of the waveform shown in Figure. 16.