Experiments in preprocessing techniques for underwater acoustic target recognition

Antriksh Dhand

A thesis presented in partial fulfilment of the requirements for the degree of

Bachelor of Engineering Honours (Software)

Supervisors:

Daniel La Mela, Thales

Dr. Dong Yuan, The University of Sydney

School of Electrical and Computer Engineering

The University of Sydney

December 16, 2024

Abstract

Underwater acoustic target recognition (UATR) is a critical task in the application of sonar systems that aims to classify objects based on their acoustic signatures. Traditionally, UATR has relied on rule-based systems and the expertise of highly-trained sonar technicians to extract and classify features from raw sonar signals. However, recent advancements in artificial intelligence, particularly the rise of deep learning, have spurred interest in automating this process.

A key factor influencing classification accuracy in machine learning models is the quality of the input data. To this end, various preprocessing techniques have been developed to enhance sonar signal quality by reducing noise and highlighting relevant features. This thesis evaluates the impact of three preprocessing techniques – normalisation, detrending, and denoising – on UATR performance, using the DeepShip dataset and a hybrid convolutional neural network-long short-term memory (CNN-LSTM) model as the experimental foundation.

Experiments with normalisation revealed minimal impact, largely due to the inherent consistency of the dataset and prior preprocessing steps, such as power spectrogram conversion. Detrending with ℓ_1 algorithms consistently reduced classification accuracy, likely due to over-smoothing and disruption of spectrogram features critical for the CNN-LSTM model. Efforts to adapt the Noise2Noise framework for denoising underwater spectrograms highlighted the challenges of dynamic underwater environments, where its assumptions could not be effectively met. Despite these limitations, masking-based denoising techniques showed promise in isolating regions of interest in spectrograms, offering a viable direction for future exploration.

This thesis underscores the unique challenges of adapting machine learning techniques to the underwater acoustic domain. The findings highlight the need for tailored preprocessing and model development to address the inherent variability and complexity of sonar data, paving the way for more robust and effective UATR systems.

Declaration

I hereby declare that this thesis is wholly my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Antriksh Dhand

December 18, 2024

Acknowledgements

I would like to express my gratitude to Thales for providing me with the opportunity to work on such a challenging project. I thank my supervisors and colleagues, Daniel La Mela, Chang Sung, and Mark Tadourian, for their guidance and support over the past six months. I further extend my appreciation to my academic supervisor, Dr. Dong Yuan, for his advice and continuous encouragement throughout this process.

I would also like to dedicate this work to those who have supported me emotionally over the past six months. To my family – Mum, Dad, and Oshin – this thesis would not have been possible without the support and foundation you have always provided me. To Jocelyn, for helping me make it through this internship despite being all the way in America: you've proven that true friendship knows no geographical bounds. To Udit, for reminding me that this is just one step in a much bigger picture. And to all my other friends, who are probably tired of hearing about my thesis – thank you for being there, listening, and supporting me from the sidelines. This work is dedicated to all of you.

"Machine intelligence is the last invention that humanity will ever need to make."

Nick Bostrom

Contents

List of Figures

List of Tables

List of Abbreviations

AE autoencoder

AI artificial intelligence

AST audio spectrogram transformer

CAE convolutional autoencoder

CNN convolutional neural network

CNN-LSTM convolutional neural network–long short-term memory

CQT constant-Q transform

DBN deep belief network

DEMON decomposition of modulated noise

DL deep learning

DSP digital signal processing

EMD empirical mode decomposition

FFT fast Fourier transform

GFCC Gammatone-frequency cepstral coefficient

GPU graphics processing unit

HMM hidden Markov model

IoU intersection over union

KNN K-nearest neighbours

LOFAR low-frequency analysis and recording

LSTM long short-term memory

MFCC Mel-frequency cepstral coefficient

ML machine learning

MLP multi-layer perceptron

MSE mean squared error

N2N Noise2Noise

N2V Noise2Void

PCA principal component analysis

PSNR peak signal-to-noise ratio

RBM restricted Boltzmann machine

RNN recurrent neural network

SAE sparse autoencoder

SNR signal-to-noise ratio

SOFAR sound fixing and ranging

SOSUS sound surveillance system

SSIM structural similarity index measure

STFT short-time Fourier transform

SVM support vector machine

UATR underwater acoustic target recognition

VAE variational autoencoder

VMD variational mode decomposition

ZCR zero crossing rate

Chapter 1

Introduction

Submarine operators detect the presence of vessels in the surrounding waters primarily through the use of SOund Navigation and Ranging (sonar), a technology that leverages the propagation of sound waves through water to analyse the presence, location, and characteristics of underwater objects [1]. Sonar works analogously to how audio recording works in air: sound is carried through the water medium as longitudinal waves with a certain frequency and energy, and this pressure is picked up by a recording device called a hydrophone. These hydrophones convert the mechanical energy of the wave into an electrical signal, which can then be analysed using a variety of methods in the field of digital signal processing (DSP) [2], [3].

Classifying an object based on its sonar signature, a task known as underwater acoustic classification or underwater acoustic target recognition (UATR), is a difficult job. This process is traditionally split into two stages: feature extraction and classification. Feature extraction focuses on identifying and isolating key attributes from raw sonar signals, while classification involves applying various models to categorise the input signals based on these extracted features (see Figure 1.1). Currently, both these tasks are being undertaken by teams of highly-trained sonar technicians around the world [4]. These teams use large databases of patterns and signals collected from known objects over the years to compare and classify a new incoming signal [1]. It is a slow and cumbersome task which at present relies on traditional rule-based learning techniques [5]. However, with recent advancements in artificial intelligence, researchers aim to automate UATR, making it faster and potentially more accurate [4], [5]. Improving this automation of UATR is the primary focus of this thesis.

1.1 Problem statement

The underwater acoustic environment poses significant challenges due to its unpredictable and noisy nature. Factors such as signal reflection, refraction, and attenuation, along with diverse noise sources like marine life, vessel traffic, and environmental disturbances, create highly variable recordings. These conditions obscure meaningful features in sonar signals, making accurate classification for underwater acoustic target recognition (UATR)

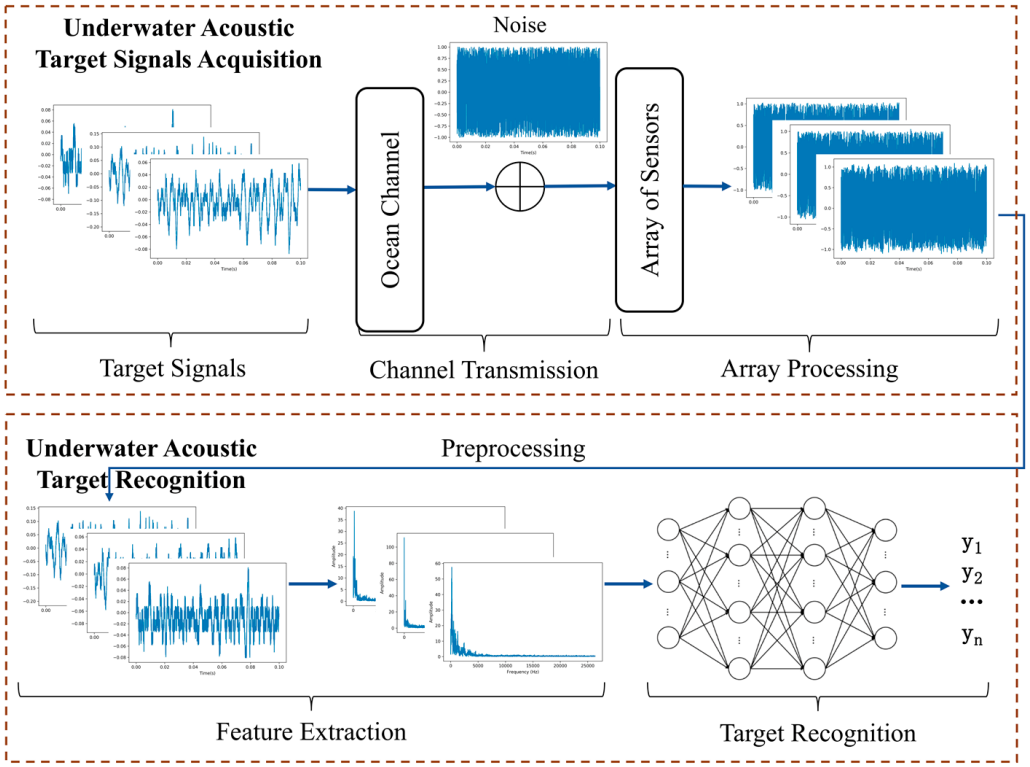


Figure 1.1: Flow chart of underwater acoustic signal acquisition and target recognition, obtained from [6, Fig. 1]

a complex task. Without effective preprocessing, machine learning models tasked with classification are often overwhelmed by irrelevant noise, resulting in reduced accuracy and reliability.

Preprocessing techniques, including normalisation, detrending, and denoising, are essential for improving the quality of sonar signals. These methods reduce noise, suppress irrelevant trends, and emphasise key signal features, enhancing the effectiveness of downstream classification tasks. Despite their importance, limited research has systematically explored the role of traditional preprocessing techniques in the context of modern deep learning-based UATR. The unique challenges of the underwater domain, such as limited labelled datasets and the inherent variability of sonar signals, demand a tailored investigation into how preprocessing can enhance feature representations for classification.

This thesis aims to address this gap by exclusively focusing on preprocessing techniques to enhance input features for UATR. Specifically, it systematically evaluates the impact of normalisation, detrending, and denoising on the performance of a benchmark CNN-LSTM classifier.

1.2 Proposed solution

In this work, normalisation techniques such as channel-based and global normalisation are assessed for their ability to standardise spectrogram inputs and improve model performance. The ℓ_1 detrending algorithm is explored for its potential to remove long-term trends, em-

phasising short-term variations that may contain critical target characteristics. Finally, denoising experiments investigate noise reduction frameworks, including the Noise2Noise methodology and masking-based techniques, to determine their efficacy in isolating relevant features in noisy underwater acoustic environments. By applying these preprocessing methods in a controlled experimental setup, this thesis quantifies their contributions to UATR accuracy and robustness, offering insights into their effectiveness and limitations for this challenging domain.

All source code for this thesis can be found at `https://github.com/antrikshdhand/thesis/tree/main`.

1.3 Outline

The structure of this thesis is as follows:

Chapter 2: Background and Literature Review This chapter provides essential theoretical background for readers, covering key aspects of underwater acoustic signal propagation, such as noise sources and their impact on signal quality, which are relevant for the denoising work explored later. The chapter also offers an introduction to artificial intelligence and concludes with a comprehensive literature review, summarising over 50 foundational and recent high-impact studies within the field.

Chapter 3: Establishing a Benchmark Performance Here, the CNN-LSTM architecture is established as the benchmark classifier model for our experiments with a detailed description of its inputs, structure, configuration, and baseline performance metrics. This architecture serves as a point of reference for evaluating all subsequent preprocessing techniques.

Chapter 4: Normalisation This chapter explores the impact of normalisation techniques on UATR performance. It evaluates channel-based and global normalisation strategies to determine their effect on standardising spectrogram inputs and improving the accuracy and stability of the CNN-LSTM model. Findings highlight the importance of aligning preprocessing techniques with the specific characteristics of the dataset.

Chapter 5: Detrending The role of detrending in UATR is investigated in this chapter, focusing on the ℓ_1 detrending algorithm. By removing long-term trends from acoustic spectrograms, the experiment aims to suppress irrelevant noise and enhance transient features critical for classification. The impact of various detrending strengths is evaluated, with results revealing the limitations and potential disruptions caused by this technique.

Chapter 6: Denoising This chapter examines denoising methods for underwater acoustic spectrograms, with a focus on the Noise2Noise framework and masking-based techniques.

It evaluates the effectiveness of these methods and highlights the challenges of adapting image-based denoising frameworks to the underwater domain.

Chapter 7: Conclusion The final chapter synthesises the findings of the thesis, summarising the contributions and limitations of the preprocessing techniques investigated.

Chapter 2

Background and Literature Review

The below section lays out the theoretical background on underwater acoustic target recognition (UATR) and conducts a broad survey of the most influential literature in the field. It also presents a detailed summary of the current publicly available datasets for use in UATR research.

2.1 Foundational principles of sonar systems

This section introduces readers unfamiliar with the subject to the foundational concepts of underwater acoustics and sonar, essential for understanding the core content of this thesis. It begins by contextualising this body of work by giving a brief overview of the historical development of acoustics and sonar technologies. Then, since sonar systems rely on the principles of underwater signal propagation, the discussion highlights key concepts and challenges when working with underwater signals such as signal attenuation, multipath propagation, and ambient noise. The section then examines active and passive sonar systems, explaining their mechanisms and the sonar equations that define their operation. Finally, it concludes by exploring several applications of sonar systems and introducing the thesis's primary focus: underwater acoustic target recognition.

2.1.1 Historical development of underwater acoustics and sonar

In 1490, polymath Leonardo da Vinci conducted an early experiment with the propagation of sound waves underwater and noted the following observation:

> If you cause your ship to stop and place the head of a long tube in the water and place the outer extremity to your ear, you will hear ships at a great distance from you. [7]

One could call this the birth of the study of underwater acoustics, as da Vinci's observation over 500 years ago is the same basic principle used in underwater sonar systems today.

Over the next four centuries, it would take the combined effort of many great minds to lay the foundations of acoustic theory. Marin Mersenne (of Mersenne primes fame) formalised the theory of vibrating strings and harmonics in 1636 [8]. Isaac Newton then pre-

sented the first mathematical theory of sound propagation in air in his 1687 work *Principia* [9]. In the early 18th century, Brook Taylor, along with Leonhard Euler, Jean d'Alembert, and Daniel Bernoulli, sought to demonstrate that any acoustic vibration could be expressed as a superposition of sinusoidal vibrations, a problem fully solved by J.L. Lagrange in 1759 [10]. Georg Ohm further contributed with his law of hearing in 1843 [11], inspiring Hermann von Helmholtz's cornerstone 1863 work *On the Sensations of Tone* [12] which focused on the perception of sound. Finally, Nobel laureate Lord Rayleigh formalised the wave equation in his seminal 1877 text, *The Theory of Sound* [13], cementing the theoretical framework for modern acoustics [14].

Ultimately, however, it was the sinking of the Titanic in 1912 that provided the financial and industrial incentives for the development of new naval navigation technologies. Canadian engineer Reginald Fessenden responded by creating the first primitive sonar system in 1914. He developed an early kind of transducer (a device capable of both sending and receiving sound) which he called the *Fessenden Oscillator*. Resembling a large loudspeaker, the device used a vibrating copper tube and steel diaphragm to produce a focused sound wave. During a test in icy waters, Fessenden lowered the oscillator into the ocean and sent out a signal. The sound waves struck an iceberg and returned as an echo just over a second later. This marked the beginnings of what we now know as *active sonar*, the underwater equivalent of radar [14], [15].

World War I further catalysed the rapid advancement of sonar technology. Extending Fessenden's work, French physicist Paul Langevin developed the first sonar system to use a crystal transducer, which sent and received sound waves using the piezoelectric effect in contrast to Fessenden's mechanical implementation. This innovation, driven by the need to detect German U-boats, represented a significant leap forward from Fessenden's earlier work [16].

World War II and the Cold War continued to propel sonar development. One of the key focuses was the development of sonar systems capable of detecting enemy submarines from long distances, even in complex and noisy underwater environments. Early on, anti-submarine warfare was dominated by the use of activate sonar systems such as ASDIC (derived from Anti-Submarine Detection Investigation Committee), which worked in a similar fashion to Fessenden and Langevin's work [17]. However, the advent of *passive* sonar, which relied on listening for sounds rather than actively emitting signals, became a major focus later on in this period. Passive sonar allowed submarines and other vessels to detect enemy movements without giving away their own location, making it a vital tool in the Cold War's "silent" naval battles.

One of the key discoveries which allowed passive sonar to work so well in performing passive sonar surveillance was the sound fixing and ranging (SOFAR) channel, discovered in 1944 by ocean scientists Ewing and Worzel. SOFAR works by taking advantage of the ocean's layered structure, where sound waves emitted at a certain depth bounce between layers of water with different temperatures and salinity. This up-and-down bending of low-frequency soundwaves allows them to travel vast distances with minimal energy loss [18].

Building on this discovery, the U.S. Navy developed the sound surveillance system (SOSUS) in the 1950s, which deployed arrays of hydrophones in the SOFAR sound channel to track Soviet submarines. The system was highly effective, enabling long-range detection in noisy environments and playing a pivotal role in Cold War anti-submarine warfare. Additionally, SOSUS arrays helped scientists study oceanic phenomena like whale calls and deep-sea currents [19].

Other technological improvements in the early stages of the Cold War included advances in array processing, where multiple sensors were used to improve signal detection, enhancing the range and accuracy of sonar systems. Additionally, algorithms were developed to filter useful sonar signals from background noise, employing methods like adaptive beamforming to focus on specific directions and improve the signal-to-noise ratio. The rise of scientific computing and the transition to digital signal processing further revolutionised the field, enabling more advanced techniques such as time-delay estimation and spectral analysis.

Progress in underwater signal processing during the 1970s and early 1980s was closely linked to advancements in general signal processing techniques. Researchers in underwater acoustics focused on improving high-resolution signal processing – that is, the ability to separate signals from sources with nearly identical frequencies or positions, such as two nearby ships or objects [14]. These efforts greatly enhanced the precision of sonar systems, enabling them to detect underwater objects more effectively, even in busy environments.

It was during the 1980s and 1990s that the field of *automatic* underwater acoustic classification, which this thesis is rooted in, truly came into its own. This shift was largely driven by the Cold War, when sonar engineers and submarine builders became acutely aware of the need to make ships "quieter." As a result, vessel-radiated noise was reduced, making it harder for sonar operators to detect man-made targets. According to Vaccaro et al. in their 1998 review of the field:

> ...the quieting of targets requires raising the threshold of detectability, which in turn increases the rate of false alarms and thus overwhelms the human operator. Automating these tasks by extracting classification features that cue the operator to relevant targets is important. [14, p. 46]

Finally, the 2000s saw significant advances in environmental applications of underwater acoustics. Many studies during this period greatly focused on the growing concerns over anthropogenic underwater noise from shipping, military sonar, and industrial activities, which pose threats to marine ecosystems by interfering with the communication and behaviour of aquatic life [20]–[22].

2.1.2 Signal propagation in the underwater environment

The most fundamental concept in underwater acoustics is sound. Being a mechanical wave, the propagation of sound relies on the density and elasticity of the medium it travels through. In turn, the density and elasticity of water is influenced by factors such as temperature, salinity, and pressure [23]. Hence, the speed of sound in water is governed by

complex relationships that incorporate these variables [24]. A simplified, empirical formula for calculating the speed of sound, c, in the ocean based on temperature T (in degrees Celsius), depth z (in meters), and salinity S (in parts per thousand) is given by [25]:

$$c = 1449.2 + 4.6T + 0.055T^2 + 1.39(S - 35) + 0.016z \tag{2.1}$$

The speed of sound in water plays a crucial role in various underwater acoustic phenomena. One of the most significant of these is refraction, which occurs when sound waves pass through layers of water with different sound speeds [7]. Refraction causes sound waves to bend as they travel through these layers, following Snell's law:

$$\frac{\cos\theta_1}{c_1} = \frac{\cos\theta_2}{c_2} \tag{2.2}$$

where θ_1 and θ_2 are the angles of incidence and refraction, and c_1 and c_2 are the sound speeds in the two layers, respectively [3]. The combination of Equations 2.1 and 2.2 reveals a crucial phenomenon in underwater acoustics: sound waves bend towards cooler regions of water. This refraction effect requires adapting sound profiles based on temperature variations, such as between geographic regions or different times of the year [3, p. 3]. Moreover, under certain conditions, acoustic refraction can create *shadow zones* – areas where sound waves cannot penetrate, resulting in signal loss [14].

Related to refraction is the concept of reflection. When sound waves encounter boundaries such as the sea surface or seafloor, part of the acoustic energy is reflected. The amount of reflection depends on the difference between the speed of sound in water and the boundary material, a value called the *acoustic impedence mismatch* [26]. Reflection is a key concept in several applications of underwater acoustics, particularly in the field of sonar imaging.

There are several other factors besides acoustic refraction which can cause signal attenuation in the underwater environment, including absorption, scattering, and geometrical spreading losses. Absorption loss occurs when acoustic energy is converted into thermal energy as it travels through seawater, with its extent depending on variables such as salinity, temperature, pH, pressure, and signal frequency. This absorption can range from around 0.07 dB/km at 1 kHz to 9.0 dB/km at 40 kHz, making high-frequency signals particularly susceptible to loss [14], [27]. Scattering is another significant source of attenuation as objects like fish, bubbles, rocks, and surface waves disperse sound energy in all directions, effectively masking the signal and complicating detection [27]. Additionally, frequency-independent spreading losses arise from geometrical spreading, where signal energy decays with distance. In shorter ranges, spherical spreading results in a decay rate proportional to $1/R^2$, while at longer ranges, cylindrical spreading causes a $1/R$ decay, where R is the distance from the source [14].

In addition to these sources of signal loss, underwater signal propagation is further complicated by *multipath propagation*. This phenomenon arises when sound waves reflect off the sea surface or seabed and refract through different layers of water, each with varying

sound speeds. These interactions cause the signal to split and travel along multiple paths from the source to the receiver. As a result, several versions of the same signal arrive at the receiver at different times, causing delay spreads and signal distortion. The extent of these delays varies depending on the environment. For example, in shallow waters (less than 10 meters deep), delay spreads can reach up to 10 milliseconds, while in deeper waters, these delays may extend beyond 300 milliseconds [14]. This variation in arrival times can complicate the detection and identification of underwater objects.

In summary, the underwater environment poses significant challenges for acoustic signal propagation due to a broad range of factors. Changes in temperature, salinity, and pressure affect the speed of sound, leading to phenomena such as refraction and shadow zones. Absorption and scattering of sound by marine life and other underwater phenomena contribute to signal attenuation, especially at higher frequencies. Multipath propagation results in delays and signal distortion, making the detection and classification of underwater objects particularly challenging. Collectively, these factors create a highly complex underwater acoustic landscape.

2.1.3 Sources of noise in underwater acoustics

One of the key challenges in underwater signal processing is noise, as it can obscure the original data and complicate accurate interpretation. In fact, due to the extreme complexities of the underwater environment (as described in the previous section), noise is generally a greater problem for signals underwater as compared to in-air signals. There are three main sources of underwater noise: vessel noise, ambient noise, and thermal noise [2]. Each source introduces distinct challenges for acoustic signal detection and analysis. This section aims to provide a comprehensive summary of these sources.

Vessel noise

There are three main sources of noise from a vessel, which, when summed, form the 'self-noise' of a sonar system [28]. Each of these have fairly characteristic frequency bands and are heavily dependent on the vessel speed [29].

Machinery noise originates from various mechanical systems on a vessel, including engines, pumps, and rotating equipment. These mechanical systems create vibrations that are transmitted to the ship's hull, which then acts as a medium for noise radiation. Machinery noise can be divided into discrete and continuous categories. Discrete noise is typically produced by components such as shafts, motor armatures, and turbine blades, resulting in a tonal spectrum dominated by fundamental frequencies and their harmonics, generated by the vibrating machinery. On the other hand, continuous noise is associated with repetitive impacts, such as explosions in engine cylinders, and the friction and turbulence occurring in pumps, pipes, and bearings. These processes create persistent noise that adds to the underwater soundscape [29], [30].

Hydrodynamic noise results from the ship's movement through water. As the ship moves, the water flow around the hull generates pressure variations, typically manifests

as low-frequency, broadband noise, encompassing a wide range of frequencies. This noise becomes more pronounced at higher vessel speeds or in rougher sea conditions, as the level of turbulence increases with both speed and environmental factors [27].

Propeller noise is another significant source of underwater noise, arising from the interaction between the propellers and the surrounding water. This noise is primarily categorised into cavitation noise and blade noise. Cavitation occurs when low-pressure areas around the edges of the propeller blades cause bubbles to form. These bubbles then collapse, producing high-frequency, broadband noise. In contrast, blade noise is produced by the rotation of the propeller blades, where factors such as the blade's thickness and the loading exerted on it generate narrow-band, low-frequency sounds. Propeller noise, particularly cavitation, is often a dominant contributor to underwater vessel noise, especially at higher speeds or under heavy loads [31].

Generally speaking, machinery noise is dominant at slow speeds (up to 10 knots), hydrodynamic noise is dominant at medium speeds (10 to 20 knots), and propeller noise begins to dominate above this [2].

Ambient noise

Ambient noise in underwater environments is generated from a range of natural and anthropogenic (human-induced) sources, creating a complex acoustic landscape. Below is a brief summary of some of the major sources of such noise; see Figure 2.1 for a more comprehensive overview of the frequency ranges of each source.

Seismic activity, such as underwater earthquakes (seaquakes) and volcanic eruptions, contributes significantly to low-frequency noise, especially below 20 Hz. These geophysical events introduce deep, powerful sounds into the ocean, which can travel long distances and influence ambient noise levels in various marine regions [32].

Shipping vessels are another major source of ambient noise, particularly within the 20-500 Hz range. As global shipping traffic has increased [34], the noise generated by distant cargo ships, primarily through propeller cavitation, has become a dominant contributor to ambient noise in this frequency range. The cumulative effect of numerous ships radiating sound into the ocean is substantial, raising the baseline level of ocean noise [28].

Hydrostatic effects, such as tidal movements, along with weather-related phenomena like precipitation, contribute to higher frequency noise, ranging from 500 Hz to 100,000 Hz. Wind, rain, hail, and the formation of bubbles from breaking waves all play roles in generating noise at these higher frequencies. Various factors, such as droplet size and shape, affect the acoustic characteristics of this type of noise, making it highly variable and dependent on environmental conditions [35]–[37].

Biological activity is a major contributor to ambient underwater noise levels. Sounds produced by marine life cover a wide frequency range [27]. In subtropical shallow waters worldwide, the dominant source of ambient noise often comes from the biological sounds of fish, dolphins, whales, and snapping shrimp [38]. The study of these sounds, known as bioacoustics, has provided valuable insights into animal behavior. For instance, it has

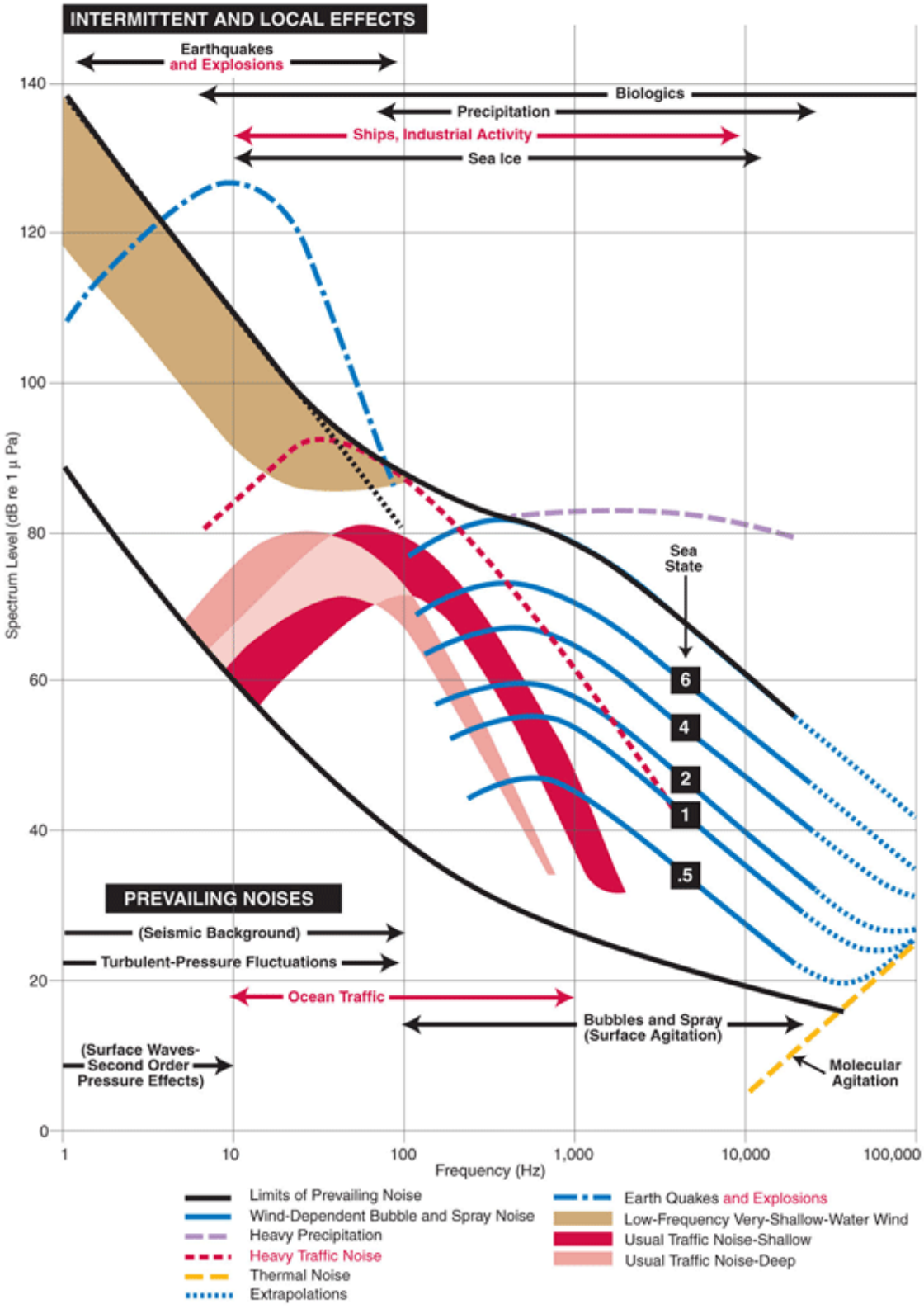


Figure 2.1: Sources of underwater noise and their approximate frequency ranges, obtained from [32], data originally from [33].

allowed us to characterise the different sounds whales use for communication, predator-prey interactions, and social bonding, such as their "whistles," "clicks," and "pulsed calls" [39]. Bioacoustics has also shed light on the mechanism behind the noise produced by snapping shrimp, with studies exploring the evolutionary and physical factors that contribute to their distinctive sound [40]–[42].

Finally, other anthropogenic activities, such as construction, offshore drilling, and underwater mining, also contribute to ambient noise. These activities can introduce a broad range of noise frequencies and intensities, exacerbating the overall noise pollution in marine environments.

Thermal noise

In the absence of ambient and vessel noise, the underlying noise level is determined by thermal noise. Thermal noise is inherent in any electrical receiving system, including sonar receivers. This noise is generated by the agitation of electrons in the electrical components, which translates to pressure fluctuations at the face of the hydrophone [2]. Mellen's expression for thermal noise is: $N_{\text{thermal}} = -15 + 20 \log f$, where f is the frequency in kilohertz. Mellen's expression highlights that thermal noise increases with frequency, impacting the sensitivity and performance of sonar systems at different operational frequencies [27]. Thermal noise dominates at frequencies around 100,000 Hz [37].

2.1.4 How does sonar work?

Sonar operates by converting acoustic energy into electrical signals, and vice versa, through specialised underwater transducers known as hydrophones [2], [16], [25]. In active sonar, the system first converts an electrical signal into an acoustic pulse using a transducer. The pulse travels through the water until it reflects off an object or the seafloor. The returning reflected sound wave, or echo, is captured by the hydrophones, which transform the acoustic energy back into electrical signals for analysis. This allows active sonar systems to determine both the range and orientation of underwater objects. Active sonar is commonly used in applications requiring high-resolution imaging, such as mapping the seafloor and identifying shipwrecks.

In contrast, passive sonar operates by listening for sounds naturally produced in the underwater environment, such as those from ship engines, propellers, biological sources, or geological activity. Passive sonar systems do not emit sound waves themselves, making them highly advantageous in military operations for stealth purposes, as they cannot be detected. However, unlike active sonar, passive systems cannot directly measure the range of an object unless multiple sensors are used to triangulate the sound source.

The performance of sonar systems, particularly in terms of detection, is typically quantified using the *sonar equations* [2], [25]. These equations estimate the signal-to-noise ratio (SNR) for both passive and active sonar systems by accounting for various factors such as source level (SL), propagation loss (PL), noise level (NL), and array gain (AG), all of which are expressed in decibels. For passive sonar systems, the SNR is calculated using the

equation:

$$SNR_{\text{passive}} = SL - PL - NL + AG \tag{2.3}$$

In active sonar systems, additional variables such as target strength (TS) and further transmission losses (TL) are considered:

$$SNR_{\text{active}} = SL - 2TL + TS - NL + AG \tag{2.4}$$

2.1.5 Applications of sonar systems

Sonar technology has a wide range of applications, many of which have seen significant advancements through the incorporation of machine learning techniques [43]. One significant application is underwater source localisation, where sonar systems detect and triangulate the position of sound sources such as submarines or marine animals using arrays of hydrophones [44]–[46]. By analysing the time difference of arrival of sound waves at different hydrophones, these systems can pinpoint a target's location, making it incredibly useful for military operations and marine biology studies. Another related application is the estimation of the direction of arrival of sound waves, which helps determine the angle from which the sound is coming [47]–[50]. Using advanced beamforming techniques in conjunction with machine learning, sonar systems can isolate specific sounds from background noise, providing crucial information in defence and navigation scenarios.

Beyond localisation and direction estimation, sonar also has more specialised applications. For instance, by analysing the sound generated by wind-driven processes such as breaking waves, it is possible to estimate wind speed over the ocean [51]. This is especially useful for remote areas where traditional meteorological instruments cannot be deployed. In bioacoustics, sonar is employed to monitor the behavior and movement of marine species, offering a non-invasive way to study endangered populations [38]–[42].

Underwater acoustic target recognition

A major application of sonar systems is underwater acoustic target recognition (UATR), which forms the central focus of this work. Simply put, UATR is the analysis of a sonar signal with the aim of determining its source. The sonar signal can either be active or passive, however, as described in Section 2.1.1 as well as in [52], most applications nowadays rely on passive sonar in order to limit self-detectability as well as harm to sea life. The techniques used for this analysis can be split into two categories: the conventional approach using classical signal processing tools and the machine learning (ML) approach.

In conventional underwater acoustic target recognition, human sonar operators are responsible for analysing real-time acoustic data to detect and classify underwater objects. The process begins as described in Section 2.1.4, with sonar systems detecting sound waves through arrays of hydrophones and converting the acoustic signals into electrical signals. These electrical signals then find their way onto two main interfaces: the broadband display and the narrowband display.

Figure 2.2: Sonar operators observing broadband and narrowband displays, obtained from [53].

The broadband display, often referred to as the "waterfall display", displays the total acoustic energy received from the hydrophone array with the latest data "cascading" from the top of the screen. This interface allows operators to detect spikes in acoustic energy which could indicate the presence of a target. At this stage, operators observe the different bearings (the direction of the signal relative to the vessel) of the incoming signal and assign it to some broad classification based on their initial impressions, such as biological sound, seismic activity, or vessel noise.

Once a general classification is made, sonar operators transition to the narrowband display, where the broadband signal is divided into its component frequencies through the use of a Fourier transform. Since each object produces distinct spectral characteristics (as discussed in Section 2.1.3), operators draw upon various conventional signal processing concepts such as harmonics and modulation patterns to analyse these frequency components in order to classify the incoming signal.

For example, the harmonic structure of mechanical noise generated by a ship's engine or propeller is distinct and recognisable due to its periodic nature, resulting in a clear pattern of fundamental frequencies and their harmonics [30]. In contrast, the random noise produced by environmental factors like wind, waves, or marine life is more dispersed across the spectrum, often lacking this periodicity. Operators also look for modulation patterns (changes in amplitude or frequency over time) that may reveal whether the sound is coming from a rotating propeller, hydraulic machinery, or other mechanical components. The operator may also apply band-pass filters to isolate frequencies associated with specific targets, such as the low-frequency analysis and recording (LOFAR) technique, which focuses on low-frequency sounds characteristic of large vessels. This in-depth spectral analysis typically takes between 30 to 60 seconds, during which operators confirm or revise their initial broad classification [53].

It is widely accepted that the job of a sonar operator is a challenging task. In high-contact environments, operators face the challenge of managing multiple potential targets simultaneously. They must continuously verify bearings and frequencies, ensuring that the sonar system maintains a reliable track on all contacts. This task is mentally demanding, and sonar watches can last for up to six hours, requiring intense concentration from operators. In addition, due to the inherent setting of the job, sonar operators never truly know that their classification is correct. These are just a few compelling reasons as to why the conventional approach to UATR continues to dominate in today's naval operations despite being much more resource-intensive. The expertise, intuition, inferential comprehension, and real-time decision making ability of sonar operators provides a level of accuracy and reliability in vessel classification that no AI system has fully replaced.

It is important to reflect on this in order to appreciate the inherent limitations of applying AI techniques like machine learning to UATR. Most modern AI methods are designed to create *rational agents* – machines that *act rationally* and "do the right thing" by making decisions that maximise some performance metric. For example, machine learning models aim to uncover underlying patterns in datasets and optimise their predictions by minimising error through a loss function. This is fundamentally different from the earlier, *human-centred* approaches to AI, which sought to build systems that could think or act like humans (Table 2.2) [54], [55][a].

This distinction matters because UATR is a field that inherently benefits from human abilities such as inference, intuition, context-driven decision-making, and problem-solving. As discussed previously in this section, the underwater acoustic environment is incredibly complex and dynamic. Ambient background noise levels can change drastically within hours, properties such as reflection, refraction, signal attenuation and multipath propagation introduce complexities with signal propagation, and the qualities of sound can differ based on numerous factors such as temperature, salinity and pressure. In such a challenging environment, thinking like a human by synthesising diverse streams of information, including past experience, environmental context, and indirect clues, to form a probable classification for an incoming signal is an almost essential part of the job, and is a skill not easily replicated by AI systems optimised solely for pattern recognition [56].

Despite this, there has been a concerted effort in recent years to develop AI systems that can perform UATR autonomously. Researchers have turned to machine learning and, more recently, deep learning, in an attempt to automate the tasks traditionally carried out by sonar operators. These AI systems attempt to classify underwater signals by training on large datasets and using either statistical models or deep neural networks for classification. While these approaches hold promise, it is worthwhile to keep in mind that they will inherently struggle to provide the adaptability, intuition, and contextual reasoning that human operators bring to the table.

[a] See Section 2.3.1 for more on the historical development of AI.

2.2 A review of traditional features for UATR

Feature extraction is the process of identifying and isolating key characteristics of the acoustic signal that can be used to effectively classify underwater targets, and is the crucial first step in any UATR system. This process can be broadly divided into two major categories: traditional signal processing techniques and modern deep learning or self-supervised methods. The former draws from well-established concepts in signal processing, often inspired by physics and human auditory perception, while the latter relies on models that can automatically learn representations from raw data. Understanding traditional techniques remains essential, as many machine learning algorithms build upon these well-established features, incorporating them as input representations. In the following section, we provide a brief overview of traditional feature extraction methods, while deep learning-based feature extraction is discussed in detail in Section 2.3.3.

Traditional feature extraction methods in signal processing are typically classified into three primary domains: time-domain, frequency-domain, and time-frequency features. Each domain focuses on different aspects of the signal to capture important characteristics. Time-domain features analyse the signal's behavior over time, capturing metrics such as amplitude, energy, and zero-crossing rate. Frequency-domain features, on the other hand, examine the signal's spectral content using transformations like the Fourier transform. Time-frequency features combine both domains, using techniques like the STFT or wavelet transforms to analyse how frequency components change over time. The following sections will delve into the most commonly used techniques within each of these categories.

2.2.1 Time-domain features

Energy-based features

Energy-based features measure the total energy of a signal within a specific time window. They are computed as the sum of squared amplitudes and are useful for characterising the signal's intensity and overall loudness. The energy E can be calculated as:

$$E = \sum_{n=1}^{N} x(n)^2 \tag{2.5}$$

where $x(n)$ is the signal amplitude at time n and N is the number of samples. Such features help in understanding the signal's strength but do not provide information on how the energy changes over time.

Zero crossing rate

The zero crossing rate (ZCR) quantifies the rate at which the signal changes sign. It is defined as the number of times the signal crosses zero within a given time frame:

$$ZCR = \frac{1}{2} \sum_{n=1}^{N-1} |\text{sgn}(x(n)) - \text{sgn}(x(n+1))| \tag{2.6}$$

where $\text{sgn}(x)$ represents the sign function. A high zero crossing rate usually indicates a noisy or high-frequency signal.

Autocorrelation

Autocorrelation measures the similarity between a signal and a delayed version of itself. It is useful for detecting repeating patterns and periodicities within the signal. The autocorrelation function $R(\tau)$ is given by:

$$R(\tau) = \frac{1}{N-\tau} \sum_{n=1}^{N-\tau} x(n) \cdot x(n+\tau) \tag{2.7}$$

where τ is the time lag.

Amplitude envelope

The amplitude envelope represents the smooth curve outlining the peaks of a signal's waveform. It is obtained by applying a low-pass filter to the absolute value of the signal. The envelope highlights the variations in amplitude over time.

Entropy-based features

Entropy measures the randomness or unpredictability of a signal. Higher entropy values indicate more complex signals. The entropy H can be calculated as:

$$H = -\sum_{i=1}^{K} p(i) \log p(i) \tag{2.8}$$

where $p(i)$ is the probability of occurrence of the i-th event and K is the number of distinct events.

2.2.2 Frequency-domain features

Power spectrum

The power spectrum shows how the power of a signal is distributed across different frequencies. It is obtained by squaring the magnitude of the Fourier transform of the signal.

The power spectrum $P(f)$ is given by:

$$P(f) = |\mathcal{F}(x(t))|^2 \tag{2.9}$$

where $\mathcal{F}(x(t))$ is the Fourier transform of the signal.

Cepstrum

The cepstrum is used to analyse the periodicity of a signal by applying the inverse Fourier transform to the logarithm of the power spectrum. It helps in identifying the pitch and other periodic structures. The cepstrum $C(t)$ is calculated as:

$$C(t) = \mathcal{F}^{-1}\{\log |\mathcal{F}(x(t))|^2\} \tag{2.10}$$

Mel spectrum

The Mel spectrum uses a scale that approximates the human ear's perception of pitch. The frequency axis is warped to emphasise lower frequencies. The Mel frequency scale is calculated as:

$$f_{mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.11}$$

where f is the frequency in Hz. This scale is useful for mimicking human auditory perception.

Mel-frequency cepstral coefficients

Mel-frequency cepstral coefficients (MFCCs) are coefficients that represent the short-term power spectrum of a sound signal on the Mel scale. MFCCs are computed by applying a discrete cosine transform to the Mel spectrum, providing a compact representation of the signal's spectral properties.

Gammatone-frequency cepstral coefficients

Gammatone-frequency cepstral coefficients (GFCCs) are similar to MFCCs but use the gammatone filter bank instead of the Mel filter bank for frequency analysis. They are useful for analysing complex audio signals and have a more accurate representation of auditory perception.

DEMON spectrum

The decomposition of modulated noise (DEMON) spectrum captures the modulation characteristics of noise signals. It is used to analyse non-stationary signals by decomposing them into components with different modulation frequencies.

LOFAR spectrum

The low-frequency analysis and recording (LOFAR) spectrum focuses on low-frequency noise analysis, which is crucial for understanding environmental and underwater noise sources. It provides a detailed representation of low-frequency components.

2.2.3 Time-frequency representations

Spectrograms

Spectrograms display the frequency content of a signal over time. They are obtained by applying the Short-Time Fourier Transform and plotting the magnitude of the spectrum. Time is on the x-axis and frequency is on the y-axis.

Wavelet decomposition

Wavelet decomposition involves analysing a signal at multiple scales using wavelets, which are waves with a finite duration and time with zero mean. Wavelet decomposition has proven to be suitable for analysing signals that contain information at different frequencies and time.

Track detection

As discussed in-depth in Section 2.1.5, narrowband sounds radiated by a ship's internal machinery or propellers typically form distinct tracks on a spectrogram. Identifying these frequency tracks can reveal critical details about the source's movement or function. Automatically detecting these tracks is the goal of *track detection*. There have been numerous algorithms devised to solve this problem, with the most recent comprehensive survey of the field being undertaken by Lampert and O'Keefe in 2010 [58]. However, given that this survey is over a decade old, it does not incorporate newer deep learning techniques that have shown promising results in track detection. To address this gap, Table 2.1 offers a brief literature review of recent advancements in the field.

2.3 Artificial intelligence for UATR

This section provides the reader with the necessary background to understand how artificial intelligence and machine learning techniques are used for UATR. It begins with a brief historical overview of the field, tracing its development over the 20th and 21st centuries. A comprehensive literature review of classification techniques follows, drawn from over 50 relevant studies. The discussion then shifts to the use of deep learning methods for the automatic extraction of features from input data. Finally, the section concludes with an overview of the datasets commonly used to train ML models for UATR.

Table 2.1: A brief literature review of track detection algorithms.

Paper	Year	Notes
Abel et al. [57]	1992	A seminal text introducing image processing techniques for track detection.
Lampert and O'Keefe [58]	2010	First comprehensive survey of track detection algorithms.
Zhang et al. [59]	2018	Employs PCA but struggles to distinguish closely spaced lines due to envelope smoothing.
Luo and Shen [60]	2019	Uses HMM and segmentation to build a track detection model with low complexity.
Han et al. [61]	2020	Presents *DeepLofargram* based on AlexNet and VGGNet, achieving robust performance at -24 dB SNR.
Li et al. [62]	2020	Extracts whistle contours of toothed whales using a residual-based neural network.
Huang et al. [63]	2021	Uses autoassociative neural network on a proprietary dataset to extract line spectrums with moderate success.
Luo and Shen [64]	2021	Uses track-before-detect technology, HMM, and Viterbi algorithm.
Ju et al. [65]	2022	Autoencoder-based architecture with added noise and delay. Promising results given small training dataset.
Li et al. [66]	2023	Proposes a novel method for extracting interference striations (characterised by parabolic patterns) using decomposition and clustering techniques.
Li et al. [67]	2023	Introduces DL framework *AINP+LR-DRNet* focusing on robustness under non-Gaussian noise. Promising results, particularly at low SNR.
Zhou et al. [68]	2023	Uses a genetic algorithm and stochastic resonance approach for enhancing track detection in low SNR environments.
Li et al. [69]	2024	Introduces *DEDAN*, a convolutional encoder-decoder network which displays promising results compared to HMM and UNet.

2.3.1 Historical development of AI

Artificial intelligence is a relatively recent addition to the scientific corpus, with its foundations laid in the mid-20th century. This section aims to provide a concise historical overview of AI and position machine learning within the broader AI landscape[b].

Early days and the advent of the perceptron

Perhaps the most intuitive way to introduce artificial intelligence is through the framework provided by Russell and Norvig. They propose that artificial intelligence can be categorised into two main schools of thought: *human-centred* and *rationality-centred* (Table 2.2). The human-centred approach seeks to build systems that mimic human thinking or behavior. This approach is closely aligned with subfields like neuroscience, artificial general intelligence, and robotics, where the goal is to replicate or simulate human cognition. In contrast, the rationality-centred approach focuses on designing agents that make optimal decisions, independent of whether they think or act like humans. This latter approach dominates modern AI research, particularly in subfields such as machine learning, where algorithms are trained to optimise decision-making based on the data they process.

Table 2.2: The four historical perspectives of artificial intelligence proposed by Russell and Norvig [55].

Human-centred approach	**Rationality-centred approach**
Think like humans	Think rationally
Act like humans	Act rationally

The earliest efforts in the field were human-centred. For example, the Turing test, introduced by Alan Turing in 1950, measures a machine's ability to exhibit human-like intelligence by determining whether its responses are indistinguishable from that of a human's [71].

Another key idea, also inspired by the workings of the human brain, was the *perceptron*, developed by McCulloch, Pitts, and Rosenblatt in the mid-20th century in an attempt to replicate the brain's decision-making processes [72], [73]. The perceptron is an abstraction of a biological neuron, which (as was only recently discovered at the time) consists of several key components: the cell body, which contains the nucleus; dendrites, which receive input signals; an axon, responsible for output; and synapses, which allow communication between neurons through chemical activation across a narrow gap between one neuron's axon terminal and another neuron's dendrite (Figure 2.3).

When the combined input signal from the dendrites is strong enough, the neuron reaches a threshold, triggering an electrical signal called an "action potential" that travels down the axon. As this action potential reaches the synapse, it causes the release of neurotransmitters

[b]For a more extensive historical recount of the field, readers are encouraged to refer to McCorduck's "Machines Who Think" [70] and Chapter 1 of Russell and Norvig's standard reference "Artificial Intelligence: A Modern Approach" [55].

across the synaptic gap, facilitating the transfer of chemical signals to the dendrites of the adjacent neuron. These neurotransmitters then bind to receptors on the target neuron's dendrites, influencing whether that neuron will also generate an action potential. Some synapses excite the receiving dendrites, increasing the likelihood of neuron activation, while others inhibit them, reducing this likelihood. This process continues as signals propagate through the neural network.

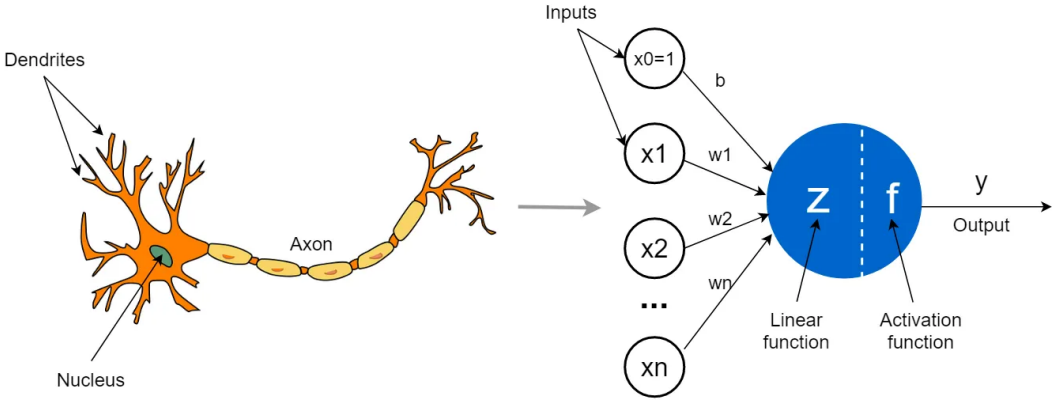


Figure 2.3: Illustration of a biological neuron and its analogue, the perceptron [74].

Mathematically, the perceptron abstracts this biological process. The inputs to a perceptron $(x_1, x_2, \ldots, x_n)$ are modelled as an input vector $\boldsymbol{x}$, where each input represents a dendrite's signal. To represent the strength of the synaptic connection, each input is assigned a weight w_i, forming a weight vector $\boldsymbol{w}$. The neuron "fires" when the weighted sum of the inputs, $\boldsymbol{x} \cdot \boldsymbol{w}$, exceeds a certain threshold determined by an activation function, f. There are many different formulations of activation functions, the simplest perhaps being the unit step function which fires when the weighted sum is non-negative:

$$a = f(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases} \tag{2.12}$$

Additionally, a bias term b_i is introduced to adjust the firing threshold, and so the perceptron's output is determined by the following equation:

$$a_i = f(\boldsymbol{w} \cdot \boldsymbol{x} + b_i) \tag{2.13}$$

Note that the perceptron either fires or does not fire – in other words, its output is binary. This binary output was initially designed for performing simple binary classification tasks, where the input vector $\boldsymbol{x}$ would be classified as either a positive or negative instance. However, a key limitation of this model is that it only works when the input data is *linearly separable*. This limitation was famously highlighted in 1972 by Minsky and Papert in their book *Perceptrons* [75], where they demonstrated that the perceptron could not solve the

XOR problem, a fundamental logic problem that requires the model to output true when the two inputs are not equal, and false when they are. The inability of perceptrons to learn non-linear relationships was a significant roadblock for early AI research and was, in-part, one of the causes for the first "AI winter" (1974–1980).

The origins of deep learning

The limitations of the perceptron in solving non-linearly separable problems such as XOR prompted the development of more sophisticated models. Researchers realised that deeper architectures, where perceptrons were layered on top of each other, could address the inability for perceptrons to capture non-linear decision boundaries. This led to the development of the *multi-layer perceptron* (MLP), which is essentially a network of perceptrons arranged in layers. An MLP consists of an input layer, one or more hidden layers, and an output layer. Each node in the hidden layers performs a weighted sum of inputs, followed by an activation function, and then passes the result to the next layer. This deeper architecture – giving rise to the field *deep learning* (DL) – allows the network to capture non-linear relationships. However, the biggest challenge that arose was figuring out how to train such networks efficiently.

The breakthrough for training MLPs came with the *backpropagation algorithm*, formalised by Rumelhart, Hinton, and Williams in 1986 [76]. This algorithm solved the problem of efficiently calculating the gradients required to adjust the weights in a multi-layer network. Backpropagation uses the chain rule of calculus to compute the gradients of the loss function with respect to each weight, allowing the network to perform gradient descent to minimise the error. This technique allowed neural networks to be trained to solve more complex, non-linear problems and was a foundational development for modern deep learning.

Another significant advancement in artificial intelligence came in the late 1980s and early 1990s with the development of the convolutional neural network (CNN), a breakthrough often accredited to LeCun and Bengio [77], [78]. CNNs use convolutional layers which apply filters (or kernels) across an input to extract local features, followed by pooling layers to reduce the dimensionality of the data. This hierarchical structure allows CNNs to capture complex patterns in data with spatial structure such as images by focusing on local regions [79, Ch. 9]. CNNs became especially effective in domains requiring pattern recognition such as image classification and object detection. For example, LeCun used CNNs to solve digit recognition tasks, notably developing the LeNet-5 architecture [80], which became the foundation of many modern applications in computer vision.

The 21st century: big data, big machines, and deep networks

The 2000s marked a significant shift in artificial intelligence and machine learning, largely due to the rise of big data. With the advent of the internet and social media, vast amounts of data became available for training machine learning models. Two of the most prominent datasets from this era include ImageNet [81] and MS-COCO [82], which became benchmarks

for computer vision tasks and helped in providing the sheer volume of training data needed for complex architectures.

A key technological enabler of these advancements was the introduction of the graphics processing unit (GPU) in the late 2000s which revolutionised the speed with which models could be trained. Compared to traditional CPUs, GPUs offered up to 70x faster training times [83], allowing researchers to explore more complex models with greater efficiency, dramatically accelerating progress across a range of artificial intelligence disciplines.

The 2010s were an incredibly transformative period for deep learning, especially in CNN architectures [84]. Several key architectures were proposed over the decade including AlexNet in 2012 [85] which introduced the use of ReLU (Rectified Linear Units) as an activation function and demonstrated the power of using GPUs for training; VGGNet in 2014 [86] which improved upon AlexNet by proposing a simpler architecture, reducing the number of parameters required to obtain feature maps and thus improving convergence time and model performance; and ResNet in 2015 [87] which tackled the vanishing gradient problem in very deep networks by introducing the *residual block*, allowing the network to skip layers during training by introducing skip connections.

The decade was not over yet; in 2017, Vaswani et al. presented the groundbreaking transformer model in their now-canonical paper *Attention Is All You Need* [88]. The key innovation in this model was the self-attention mechanism, which allowed the model to focus on different parts of the input sequence dynamically, enabling it to capture long-range dependencies more effectively than traditional RNNs and LSTMs. This made it incredibly popular for natural language processing models such as Google's BERT [89] and OpenAI's GPT-3 [90].

Soon, transformers expanded into other domains, including computer vision. The 2020 paper *An Image Is Worth* 16×16 *Words* [91] introduced the vision transformer model which adapted transformer architectures to image processing tasks by splitting images into patches (analogous to words in text). This paper showed that transformers can surpass traditional CNN architectures, however only on the condition that the models are trained on incredibly large datasets (14-300 million images).

In summary, the field of artificial intelligence has undergone a rapid evolution in just a short period of time, from early innovations such as the perceptron and backpropagation to the emergence of deep learning architectures like CNNs and, more recently, transformers.

2.3.2 A review of classification techniques

This thesis is rooted in the subfield of machine learning, a branch of AI that is generally divided into three main paradigms: *unsupervised learning*, where the system identifies patterns and relationships given data without labelled outputs; *supervised learning*, where the system is provided both input data and corresponding labels and learns to map inputs to outputs; and *reinforcement learning*, where the system learns by interacting with an environment and receiving feedback in the form of rewards or penalties. Supervised learning can further be divided into regression or classification based on whether the goal is to

predict a continuous quantity or discrete class labels. UATR primarily employs supervised classification, where the goal is to classify a given sonar signal into a certain vessel type.

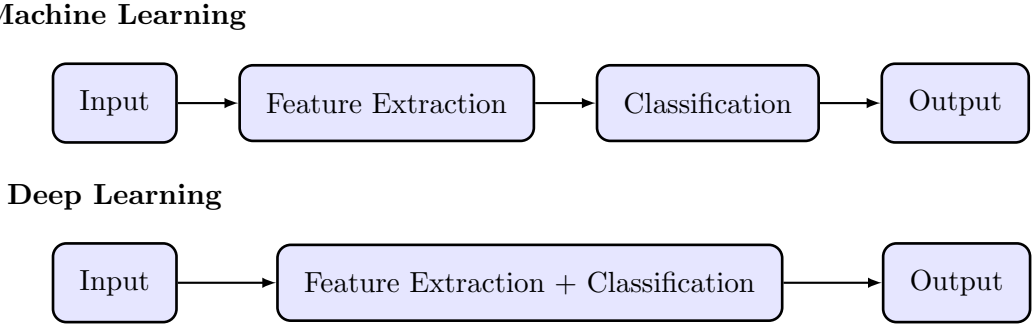


Figure 2.4: A simplified comparison between machine learning and deep learning architectures, recreated from [5].

There are many types of classifier algorithms (Figure 2.5). However, perhaps the most important distinction to make here is the difference between using traditional machine learning models and modern deep learning models (Figure 2.4). Traditional ML models, such as SVM or KNN, typically rely on manual feature extraction. These models depend on domain expertise to hand-engineer features from sonar signals, which are then fed into a classification algorithm. However, this feature engineering can be rigid and limited, as it struggles to capture complex variations and nuances in highly dynamic underwater environments, leading to reduced reliability when conditions change unpredictably [56, p. 2]. Deep learning architectures, on the other hand, automatically learn relevant features from the raw data through using neural networks that have many hidden layers and more complex connections between them – hence the term deep learning [5]. Unlike traditional ML models, DL systems can process data in an end-to-end manner, eliminating the need for manual feature extraction. Additionally, DL models are more robust when working with large datasets, as they can leverage the sheer volume of data to improve accuracy and generalise better to new, unseen data. However, while DL models outperform traditional ML models in feature extraction and scalability, they require extensive labelled datasets for training and can suffer from overfitting when data is scarce.

In this section, we aim to review several key classification techniques used in UATR, ranging from traditional statistical methods to more modern deep learning approaches. In total, over 300 journal articles, conference papers, theses, preprints, and books were collected from various sources including Google Scholar, SemanticScholar, and The University of Sydney library. These were then sorted by citation count to determine the impact of the work. The top pieces of research were then read, reviewed, and summarised for the following literature review. A summary of the papers reviewed can be found in Table 2.3.

Multi-layer perceptron

The first significant work on using neural networks for sonar signal classification was presented by Gorman and Sejnowski in 1988 [92]. Their neural network model aimed to

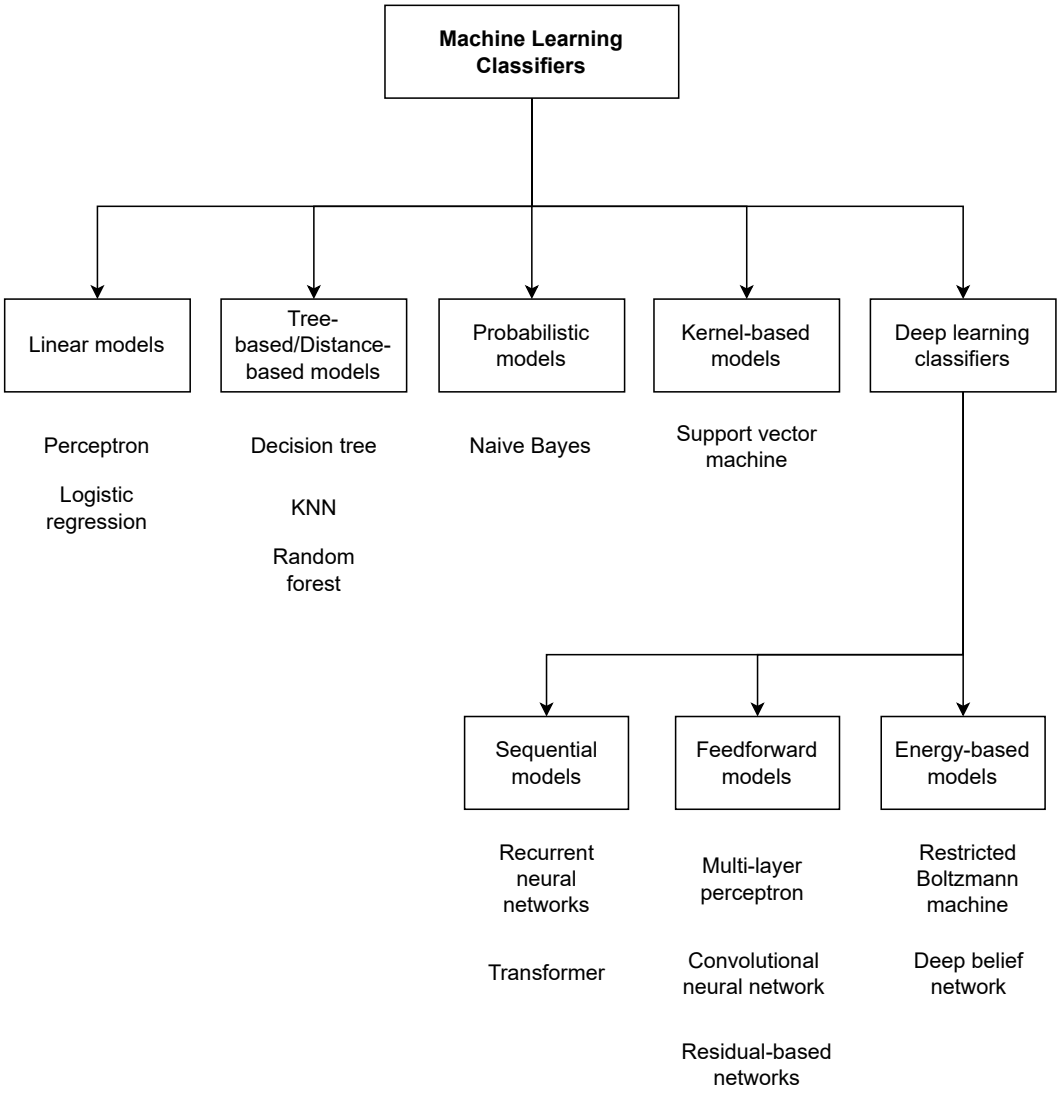


Figure 2.5: Categorisation of well-known machine learning classifiers.

Table 2.3: Summary of high-impact papers in the field of underwater acoustic target recognition.

Paper	Year	Features	Architecture	Dataset
Gorman and Sejnowski [92]	1988	Spectral envelope	MLP	Proprietary
Chin-Hsing et al. [30]	1998	Wavelet transform	MLP	Proprietary
Kamal et al. [56]	2013	Spectrogram	DBN	Proprietary
Wang and Zeng [93]	2014	Hilbert-Huang transform, Bark-wavelet analysis	SVM	Proprietary
Sherin and Supriya [94]	2015	MFCC	SVM	Proprietary
Yue et al. [95]	2017	MFCC	CNN, DBN	Proprietary
Cao et al. [96]	2019	CQT	CNN	Proprietary
Liu et al. [97]	2021	3-D Mel spectrograms	CRNN	ShipsEar
Hong et al. [98], [99]	2021	Log-Mel spectrograms, MFCC, CCTZ	Residual-based	ShipsEar
Doan et al. [100]	2022	-	Dense-CNN	Proprietary
Feng and Zhu [101]	2022	Mel spectrogram	Transformer	ShipsEar, DeepShip

differentiate between sonar echoes from a metal cylinder and a similarly shaped rock using a dataset of 208 recordings. The network featured 60 input neurons, 2 output neurons, and employed a sigmoid activation function. The experiments explored networks with no hidden layer and those with a single hidden layer (ranging from 2 to 24 units). The spectral envelope was the primary input feature, and after 300 epochs, the network achieved up to 90.4% accuracy, outperforming a KNN classifier's 82.7%. The study also highlighted that neural networks could perform comparably to human sonar operators and achieve optimal performance without requiring prior statistical knowledge of the signals.

Building on this, Chin-Hsing et al. [30] expanded the feature set for the MLP model by using the fast Fourier transform (FFT) to calculate the average power spectral density and applying wavelet transforms to extract tonal features. The neural network had 256 input units, 64 hidden units, and 2 output classes. The dataset, consisting of signals from three ships at varying speeds, demonstrated a significant accuracy improvement when wavelet-transformed features were used, reaching 96% compared to 80% with only power spectral density.

Support vector machine

The support vector machine, introduced by Cortes and Vapnik in 1995 [102], is a widely-used classification algorithm that aims to find the *maximum margin hyperplane*, which separates two classes with the largest possible margin. This maximisation ensures the classifier is more robust and generalises well to unseen data. The basic decision rule involves finding the projection of an unknown vector $\boldsymbol{u}$ onto a weight vector $\boldsymbol{w}$, with the decision boundary given by:

$$\sum_i \lambda_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{u}) + b \geq 0 \tag{2.14}$$

where λ_i are Lagrange multipliers, $\boldsymbol{x}_i$ are training samples, and y_i are class labels. To handle non-linear data, SVM uses a *kernel trick*, which computes the dot product in a higher-dimensional feature space without explicitly transforming the data, enabling efficient non-linear classification.

As a result, a significant body of research has focused on tuning SVM parameters, such as choosing appropriate kernels and kernel parameters for a given application. For example, in their 2015 paper, Sherin and Supriya [94] focused on optimising SVM parameters for underwater target classification using the BAT algorithm [103]. The dataset included 114 acoustic signals from four classes – humpback whale, sea lion, ship, and boat – processed using MFCCs for feature extraction. The hybrid kernel approach, which combines linear, polynomial, and radial basis functions, outperformed traditional methods such as particle swarm optimisation, offering a 4% improvement in classification accuracy.

Deep belief networks

Kamal et al. explore the use of deep belief networks for classifying vessel noise [56]. The authors use spectrograms created from a proprietary dataset of 40 vessel recordings as input

features for the DBN. Each RBM in the DBN is trained in an unsupervised manner, with progressively abstract features extracted at each layer. Afterward, supervised fine-tuning with backpropagation is performed over 2000 epochs to optimise the model. The DBN achieved a classification accuracy of 90.23% demonstrating robust performance in noisy environments. The results highlight the model's ability to capture deep, abstract features and its effectiveness in handling complex ambiances, making deep learning approaches like DBNs a strong candidate for underwater target recognition.

Yue et al. [95] evaluate several approaches for classifying underwater acoustic targets, comparing traditional methods like WNDCHRM with classical machine learning models such as MFCC-SVM, and deep learning models like FFT-CNN and FFT-DBN. Using World War II underwater recordings of cruisers, submarines, and torpedoes, the study finds that deep learning models outperform the traditional techniques, with FFT-DBN achieving the highest accuracy at 96.99%, followed by 94.75% for FFT-CNN, 92.15% for WNDCHRM and 86.6% for MFCC-SVM. The paper emphasises the superior performance of deep learning but notes the challenge posed by the lack of large, labelled datasets in underwater acoustics.

Convolutional neural networks

In his 2019 paper [96], Cao proposed a novel method for classifying underwater targets using CNNs combined with second-order pooling (SOP) to enhance feature extraction from CQT representations of radiated acoustic signals. The SOP was designed to capture temporal correlations of different CNN filters, improving classification accuracy by learning co-occurrences of CNN features across frequency subbands. Results showed that the proposed CNN-SOP method outperformed traditional CNN models using max-pooling, with an 8% improvement in classification accuracy over state-of-the-art methods. Additionally, the paper found that the CQT features were more effective than STFT features, yielding better performance with CNN architectures.

Doan et al.'s 2022 paper [100] uses a DenseNet-based deep convolutional neural network, named UATC-DenseNet. The dataset consists of underwater acoustic signals captured by passive sonar in a surveillance system, featuring 12 target classes. The UATC-DenseNet model is designed to process 1D time-series acoustic data by stacking convolutional blocks for feature extraction and employing dense skip connections to facilitate gradient flow and avoid vanishing gradients. The model also incorporates advanced pooling and activation techniques, including exponential linear units (eLU). Through several experiments, the authors demonstrate that larger kernel sizes and deeper networks improve classification accuracy, achieving a classification rate of 99.25% at 0 dB SNR.

Recurrent neural networks

The 2021 paper by Liu et al. [97] proposes the use of a convolutional recurrent neural network (CRNN) for UATR. The authors focus on using a 3-D Mel-spectrogram as the main feature for classification, as it captures both the spectral and dynamic properties of the signal. Data augmentation techniques, including SpecAugment [104], time stretching,

pitch shifting, and noise addition, were applied to enhance the dataset and improve the model's generalisation. The CRNN architecture combines CNNs for local feature extraction and LSTM layers for capturing temporal dependencies in the signal. Tested on the ShipsEar dataset, the CRNN-9 model achieved superior accuracy compared to standalone CNN and LSTM models, with results showing the CRNN-9 achieving a top classification accuracy of 92.8%.

Residual-based networks

The journal article [98] and conference paper [99] presented by Hong et al. propose a residual-based network for UATR. The authors use the ResNet18 architecture, which is a deep residual network with 18 layers, in tandem with features such as log-Mel spectrogram, MFCCs, and a combination of chroma, contrast, Tonnetz, and zero crossing rate (CCTZ) to feed into the network. These features are then processed using SpecAugment [104] to improve data augmentation. The model was evaluated on the ShipsEar dataset and achieved an impressive classification accuracy of 94.3%.

Transformer-based networks

Feng and Zhu's 2022 paper introduced a new model, the *UATR-Transformer* [101], which applied the transformer model to the field of UATR. The core of the model is built on a transformer architecture inspired by the Tokens-To-Token Vision Transformer. The model processes Mel-spectrogram inputs into T-F tokens which are then processed by a multi-head self-attention layer and further encoded through transformer blocks. For classification, the model replaces the traditional [CLS] token with a novel tokens pooling classifier, which jointly considers relationships between time frames and frequency bins in the spectrogram. The UATR-Transformer achieves superior performance compared to other models with an accuracy of 96.9% on ShipsEar and 95.3% on DeepShip. Additionally, the transformer-based architecture demonstrates better computational efficiency, handling the prediction tasks faster than CNN-based models due to its token embedding strategy.

2.3.3 A review of deep features

The term "deep features" refers to representations that are automatically extracted from input data using deep learning techniques. These features, unlike traditional signal processing methods (Section 2.2), are not hand-crafted by domain experts but are rather learned by the model itself during training. This section delves into the most prominent feature extraction mechanism in modern deep learning literature – encoder-decoder networks – and specifically focuses on a subset of these networks called autoencoders (AEs).

Encoder-decoder networks

At its core, an encoder-decoder network is a neural network architecture designed to take some input data, compress it into a simpler form, and then reconstruct the original data

from that compressed version. Popularised by Hinton and Salakhutdinov in their seminal 2006 paper *Reducing the Dimensionality of Data with Neural Networks* [105], these networks are made up of two main parts: the *encoder* and the *decoder*. These parts are typically connected through a middle layer known as the *latent space* or *bottleneck* (Figure 2.6).

The encoder's job is to take the input (often an image) and pass it through multiple layers of the network, gradually reducing its size and complexity. The idea is that the encoder learns to pick out the most important features of the input and represent them in a more compact form. This compressed representation is what is stored in the latent space. The decoder then works in reverse. It starts from the compressed information in the latent space and tries to reconstruct the original input data. Essentially, it attempts to "fill in the blanks" and generate a complete version of what the input should look like based on the compressed features the encoder identified.

This encoder-decoder structure is extremely useful in feature extraction because it forces the network to learn what aspects of the input are most important. Since the encoder must compress the data, it has to decide which features are essential for accurately reconstructing the input, making it a powerful tool for unsupervised learning tasks like dimensionality reduction, data denoising, or even generating new data.

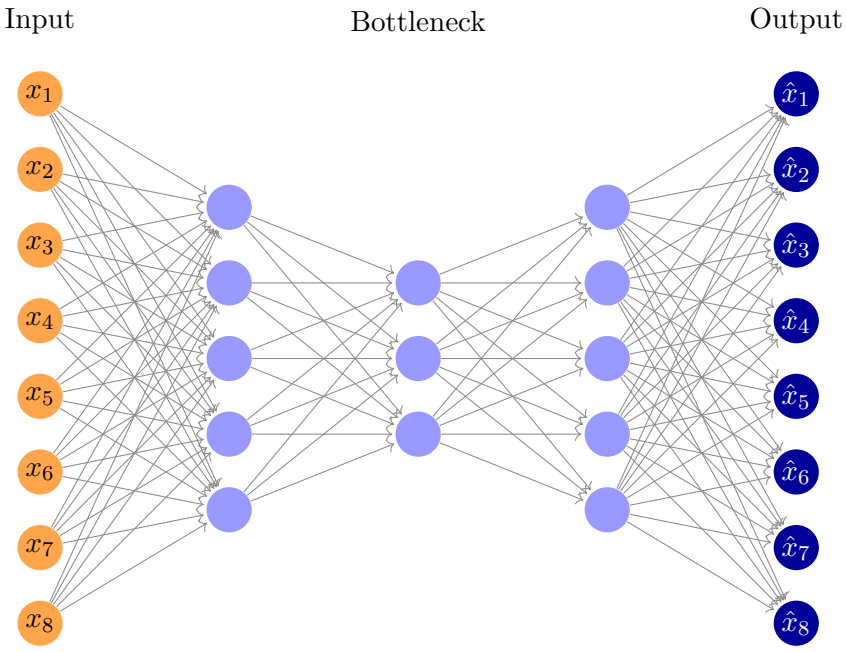


Figure 2.6: A generic encoder-decoder network.

A majority of successful encoder-decoder networks use either fully-connected layers or convolution blocks as their layers. Fully-connected layers (also called dense layers) connect every neuron in one layer to every neuron in the next. These layers are typically used in the latent space and decoder stages of the network to help map learned features from one dimension to another. Fully-connected layers are essential when a model must learn complex, global patterns, but they tend to be less efficient for tasks like image processing, where local features are more relevant.

It was due to the great computational complexity of dense layers that *convolutional blocks* were conceived, and now form the backbone of many state-of-the-art encoder-decoder networks such as UNet [106] and SegNet [107] (Figure 2.7). These blocks consist of several key components:

1. Convolution layer: applies the convolution operation to extract features.
2. Pooling layer: reduces the spatial dimension of the feature maps.
3. Activation layer: applies a non-linear activation function like ReLU.
4. Batch normalisation: stabilises and accelerates the training process by normalising activations.

The convolution operation is the core of a convolution block. As mentioned in Section 2.3.1, the idea of using convolutions for spatial feature extraction was proposed by LeCun in the 1990s [78] and involves sliding a small filter over the input data to detect patterns, such as edges or textures, by computing dot products between the kernel and overlapping regions of the input. Each filter is capable of detecting a different feature, and as the network deepens, it captures increasingly complex patterns. Convolutional layers reduce the need for fully connected layers because they preserve spatial relationships in the data.

Pooling follows the convolution operation in the block. Pooling is a downsampling technique used to reduce the dimensionality of the feature maps while retaining important information. The most common form is max-pooling, where a window slides over the input feature map and retains only the maximum value from each window, reducing the spatial size of the data. Pooling layers help decrease the computational load and prevent overfitting by summarising the presence of features in a region rather than their exact position.

After pooling, it is common to have an activation function in the convolutional block to introduce non-linearity into the network, enabling the model to learn more complex functions. One of the most commonly used activation functions in convolutional layers is the ReLU (Rectified Linear Unit), which is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{2.15}$$

Batch normalisation is an optional but extremely common final layer in a convolutional block. Batch normalisation helps improve the training speed and stability of deep neural networks through normalising the activation output of a layer, ensuring that the data distribution remains consistent throughout the network. This reduces what's known as *internal covariate shift*, where the distribution of activations changes during training making optimisation more difficult. Batch normalisation allows higher learning rates and reduces sensitivity to the initialisation of parameters. Proposed in a 2015 paper [108], batch norm is applied as:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.16}$$

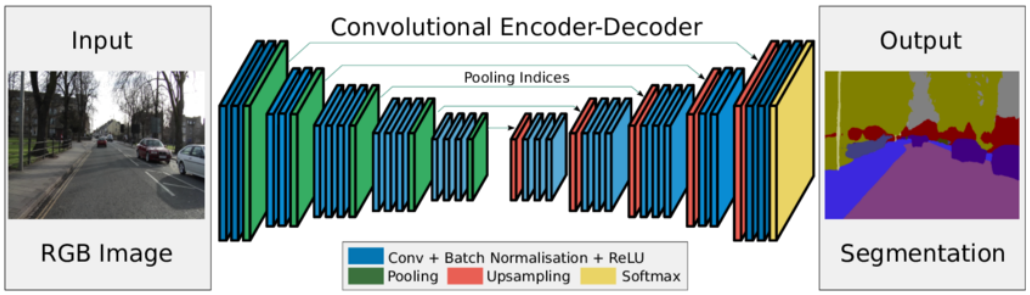


Figure 2.7: The SegNet architecture, obtained from [107].

where x is the input, μ is the mean, σ^2 is the variance, and ϵ is a small constant to prevent division by zero.

The combination of the above operations allows the network to progressively detect more complex patterns in the input data, with each convolutional block learning higher-level features from the previous block's output.

Autoencoders

An autoencoder is a specialised form of an encoder-decoder network designed primarily for unsupervised learning. Unlike other encoder-decoder networks, autoencoders do not have a classifier at the end of the decoder. Instead, they aim to minimise the reconstruction loss when reconstructing the input data from the compressed lower-dimensional latent space. As a result, the encoder is forced to learn which parts of the data are the most important, and the latent space becomes a good lower-dimensional characterisation of the original input [109]. Autoencoders are widely utilised in tasks such as data dimensionality reduction (acting as a non-linear PCA), denoising, unsupervised feature learning, and even transfer learning, where the learned representations are transferred to other models [79].

While there are many variations of autoencoders, there are three key types which are most prominent in the literature: the convolutional autoencoder (CAE), the sparse autoencoder (SAE), and the variational autoencoder (VAE). A brief review of the most impactful autoencoder-related literature has been provided in Table 2.4.

Convolutional autoencoder CAEs are built upon the architecture of CNNs, but, unlike traditional CNNs, do not end with a classification layer. Instead, the decoder simply reconstructs the original input. This makes CAEs highly effective for unsupervised learning tasks such as image denoising and anomaly detection. The convolutional layers help in capturing local spatial patterns, making CAEs ideal for handling image or time-series data.

Mathematically, for a given input image X, the encoder compresses the input into a latent representation Y in a lower-dimensional space. This can be represented as

$$Y = \text{encoder}(X) = \sigma(X * W + b)$$

where W is a set of 2D convolutional filters (weights), $*$ is the 2D convolution operator,

b is the bias term (a vector corresponding to the number of filters), σ is an activation function (e.g., ReLU), and Y is the encoded latent representation. Similarly, the decoder reconstructs the input from the latent representation:

$$G = \text{decoder}(Y) = \sigma(Y * \tilde{W} + \tilde{b})$$

where $\tilde{W}$ is a set of transposed convolutional filters (weights used for deconvolution), $\tilde{b}$ is the bias term for the decoder, and G is the reconstructed output.

The goal of the CAE is to minimise the reconstruction error, which can be achieved using a cost function such as the mean squared error:

$$E(\theta) = \frac{1}{n} \sum_{i=1}^{n} \|G_i - X_i\|^2$$

where n is the number of training samples, X_i is the i-th input sample, and G_i is the corresponding reconstructed output. The CAE uses backpropagation to minimise this error, which requires computing the gradients of the cost function with respect to the encoder and decoder weights. For the encoder weights W, the gradient is:

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial Y} \cdot \frac{\partial Y}{\partial W}$$

For the decoder weights $\tilde{W}$, the gradient is:

$$\frac{\partial E}{\partial \tilde{W}} = \frac{\partial E}{\partial G} \cdot \frac{\partial G}{\partial \tilde{W}}$$

By iteratively updating the weights using these gradients, the CAE learns to minimise the reconstruction error.

Sparse autoencoder SAEs introduce a sparsity constraint on the hidden layers of the network, which ensures that only a small number of neurons are active at any given time. This leads the model to learn compact, efficient representations of data, often uncovering more interpretable features. Earlier methods to impose sparsity, such as KL-divergence, aimed to control the average neuron activation relative to a predefined sparsity level [109]. However, more recent approaches like Dropout, introduced by Hinton and Srivastava [110], [111], enforce sparsity more intuitively by randomly deactivating neurons during training, improving generalisation and preventing co-adaptation [112].

Variational autoencoder While VAEs share the same encoder-decoder structure, they are designed for generative tasks, where the goal is to generate new data samples rather than merely reconstruct input data. VAEs introduce a probabilistic element to the latent space, learning a distribution over latent variables. Since this paper does not focus on generative models or data generation techniques, the discussion of VAEs will be limited, though they remain a significant area of research in unsupervised representation learning.

Table 2.4: A brief literature review of papers using autoencoders for UATR.

Paper	Year	Architecture	Feature	Notes
Bengio et al. [113]	2013	-	-	Review of key deep feature learning techniques.
Xu et al. [114]	2016	SAE	Image patches	Pioneering work using stacked SAEs for nuclei detection in breast cancer histopathology images.
Cao et al. [50]	2016	SAE	Spectrogram	Applied SAE with KL-divergence to enforce sparsity for feature extraction, followed by stacked SAE for classification.
Cao et al. [96]	2018	SAE	Shaft frequency, WPCE	Built on previous work, integrating features from multiple domains to improve accuracy by 5%.
Chen and Shang [115]	2019	CAE	Spectrogram	Trained CAE on unlabelled noisy samples, followed by fine-tuning with labelled data for classification.
Irfan et al. [116]	2020	CAE	Underwater images	CAE embedded with a softmax classifier for underwater image classification using Fish4Knowledge and ImageNet datasets.
Irfan et al. [117]	2021	CAE	Mel spectrogram	Introduced the DeepShip dataset and benchmarked it using a CAE model.

2.3.4 A review of datasets

AI techniques attempt to mimic the human capacity for inductive learning; that is, machine learning relies on learning from example [118]. The more data you feed into ML architectures, the better the chance that the program will pick up on the inherent underlying patterns and trends used to correctly classify new examples. Conversely, the program will have a limited accuracy if there are limited examples to learn from [5]. This property is particularly pronounced in deep learning architectures which require an even greater number of training data points in order to reach the desired accuracy and generalisation performance [119].

Unfortunately, one of the central challenges for AI researchers in UATR is the limited availability of suitable data, and this is for multiple reasons:

1. Collecting real-world underwater data is resource-intensive and costly, requiring significant time, equipment, logistical coordination, and even legal clearance [117]. It often involves complex setups and the organisation of maritime missions [3].

2. The curation and annotation of underwater acoustic data is much more specialised compared to other types of data, such as images. While anyone can label images, only trained sonar operators have the expertise to correctly label underwater acoustic signals.

3. Many underwater acoustic recordings are classified due to defence-related concerns, especially regarding military vessels. This restricts the availability of certain datasets for public use [3], [117]. For example, the military recordings collected by Zak from five Polish warships in 2008 remain unavailable due to their sensitive nature [29].

4. Underwater recordings are subject to a variety of environmental factors that can affect data quality, such as the engine regime of vessels, local recording conditions, and the acoustic propagation characteristics of the ocean [120]. Further factors include time of day, season, geographic location, sensor type, depth, and environmental conditions like pressure [3], [121]. Additionally, issues such as weak target signals, non-uniform intensities, and reverberation can further complicate data collection and processing [6]. High-quality recordings are challenging to capture due to propagation losses – such as expansion, absorption, and boundary losses – which make the process time-consuming and energy-intensive [6].

Despite these obstacles, researchers have managed to develop several key datasets to support work in underwater acoustic target recognition. Below we have traced out the evolution of these datasets, beginning with foundational private datasets before moving on to ShipsEar [120], DeepShip [117], and two recent contributions in 2024, QiandaoEar22 [122] and Oceanship [123].

Foundational private datasets

Most datasets used in underwater acoustics research have traditionally been private, often collected by the researchers themselves. Historically, much of the research in this domain has focused on the characterisation of noise generated by vessels to better understand the impact of various components on sonar signals. One of the foundational studies in this area was conducted by Arveson and Vendittis in 2000, who recorded and analysed the noise radiated from a US coal carrier, identifying the noise contributions of the diesel generator and propulsion-related sources [124]. Similarly, Pricop et al. deployed three hydrophones in the Black Sea to capture ship noise and examined the influence of engines and generators on the vessel's acoustic output [125]. Roth et al. recorded hours of noise data produced by an icebreaker operating in the Arctic, distinguishing between periods of transit and various ice-breaking activities [126]. Among the largest privately collected datasets in this field is the work of McKenna et al. who in 2012 recorded and analysed the acoustic emissions from 29 freighters traversing the Santa Barbara Channel, producing a dataset with a total duration of 422 minutes [127].

However, none of the data recorded in these studies is available publicly. Excluding the work by McKenna et al., all datasets were quite small and would only work well for traditional digital signal processing approaches. Furthermore, most of these studies were conducted for purposes other than UATR, such as noise pollution monitoring. Therefore, the data might not be structured in a way that is useful for vessel classification.

ShipsEar

ShipsEar [120] was published by Santos-Domínguez et al. in 2016 and contains 78 recordings of 39 ships in 11 vessel classes, spanning a duration of 162 minutes and 18 seconds. The authors also included 12 recordings of various ambient noise sources with the view that it would be useful to help discriminate between true signals and background noise, bringing the total dataset length to 181 minutes and 18 seconds. Each recording lasts between 15 seconds to 10 minutes. A detailed breakdown of the database can be found in Table 2.5.

Recordings were made off the Spanish Atlantic coast in northwest Spain in autumn 2012 and summer 2013. This location was chosen for its "intensity and variety of traffic" given that it is one of the largest fishing ports in the world and also experiences heavy passenger and goods traffic. This allowed the authors to record sonar signals from a variety of vessel classes such as motor boats, sail boats, fishing boats, dredgers etc. The authors also note that because the recordings are from the real environment (as compared to synthetic datasets), the dataset inevitably contains man-made and natural background noise such as from "waves crashing against the port infrastructure" and "occasional vocalisations by marine mammals".

Each recording was made with a DigitalHyd SR-1 recorder with a sampling rate of 52,734 Hz and a sampling resolution of 24 bits. The bottom of the hydrophone was connected to an underwater buoy to ensure verticality, while the top of the hydrophone was attached to a surface buoy for easy recovery. For depths over 10m, three hydrophones were

Table 2.5: Summary of the ShipsEar dataset

Vessel class	Ships	Recordings	Total recording time (min:s)
Dredger	1	5	4:22
Passengership	7	30	71:08
Tug boat	2	2	3:26
Ocean liner	4	7	17:02
Motor boat	9	13	18:34
Ro-ro	3	5	15:13
Trawler	1	1	2:43
Pilot ship	1	2	2:18
Sail boat	2	4	6:48
Mussel boat	5	5	12:10
Fish boat	4	4	8:34
Ambient noise	–	12	19:00
Total	**39**	**90**	**181:18**

Table 2.6: Experimental classes created for practical use of the ShipsEar dataset

Class	Vessel classes	Recordings
A	Fishing boats, trawlers, mussel boats, tugboats, and dredgers	17
B	Motor boats, sailboats, pilot boats	19
C	Passenger ferry	30
D	Ocean liners and ro-ro ships	12
E	Background noise	12

placed at different depths and with different gains to maximise the dynamic range of the recording (Figure 2.8).

The database can be viewed at `https://underwaternoise.atlanttic.uvigo.es`, where each recording comes with a large amount of metadata including the gains and depths of hydrophones used to record the signal, the approximate distance between the hydrophone and the target vessel, as well as any relevant meteorological data.

Despite its limitations, such as its relatively small volume and class imbalance, the ShipsEar dataset remains a significant contribution to the field of UATR. With around 3 hours of sonar recordings, it provided a foundation for machine learning and signal processing approaches, though not ideal for deep learning models which typically require larger datasets to avoid overfitting. The class imbalance issue — where the largest class (Passengerships) has approximately 31 times more data than the smallest class (Trawler) – increases this challenge. This is why Santos-Domínguez et al. suggested grouping the original 11 vessel classes into 5 experimental categories; these have since become the standard for research using the dataset (Table 2.6).

For over half a decade, ShipsEar remained the largest publicly-available labelled un-

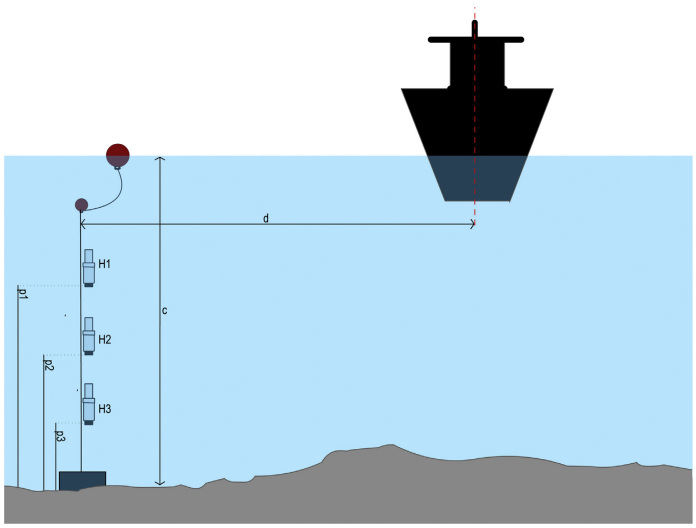


Figure 2.8: The recording setup used to create the ShipsEar dataset, obtained from [120, Fig. 1]

derwater acoustic dataset, enabling great progress in UATR research and serving as a benchmark for classification tasks in the underwater environment. However, as newer, more extensive datasets have become available, the role of the ShipsEar dataset has become more supplementary than primary in the development of modern UATR systems.

DeepShip

DeepShip [117], published by Irfan et al. in 2021, consists of 613 recordings of 265 ships, categorised into four vessel classes, with a total duration of 47 hours and 4 minutes. Each recording ranges from 6 seconds to 25 minutes in length. A detailed breakdown of the database is shown in Table 2.7.

The recordings were collected over more than two years, from May 2016 to October 2018, in the Strait of Georgia, a busy shipping route located between Vancouver Island and British Columbia, Canada. This area was selected because of its high vessel traffic, allowing the researchers to capture a diverse set of ships across multiple classes.

Each recording was captured using an icListen AF hydrophone at a sampling rate of 32 kHz, with the depth of the hydrophone ranging between 141 and 147 meters below sea level (Figure 2.9). Unlike the ShipsEar dataset, which utilised up to three hydrophones,

Table 2.7: Summary of the DeepShip dataset

Vessel class	Ships	Recordings	Total recording time (min)
Cargo	69	110	640
Passengership	46	193	742
Tanker	133	240	765
Tug	17	70	677
Total	**265**	**613**	**2824**

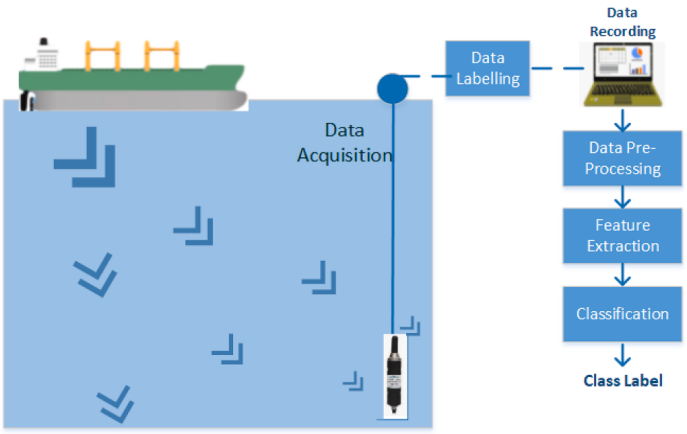


Figure 2.9: Setup used for DeepShip dataset audio collection, obtained from [117, Fig. 2].

DeepShip used a single hydrophone. To ensure high-quality data, Irfan et al. only recorded when a single vessel was within a 2 km radius of the hydrophone. The dataset is publicly available for download at `https://github.com/irfankamboh/DeepShip`.

At present, DeepShip is considered the most comprehensive and authoritative benchmark dataset for UATR research. It contains nearly seven times the number of recordings as ShipsEar and over 15 times the recording duration, making it an extremely valuable resource for training deep learning models in UATR. Its large volume of data was pivotal in allowing deep learning approaches to gain traction in this field, and it provided a common baseline for researchers to compare their results.

One advantage of DeepShip over ShipsEar is that it does not suffer from the same class imbalance problem. The recording times for each vessel class are relatively uniform, which makes it more suitable for classification tasks.

However, a limitation of DeepShip is the restricted nature of its vessel classes and the lack of detailed metadata for individual ships. The four vessel classes – Cargo, Tug, Passengership, and Tanker – are broad, and the differences between certain categories, like Cargo and Tanker, may be subtle, as the passive sonar signals generated by their mechanisms are not significantly distinct. This can make classification between these classes challenging and may limit the dataset's generalisability.

QiandaoEar22

With the issue of data availability in mind, two new underwater acoustic datasets have been published in 2024: QiandaoEar22 and Oceanship. These are both very recently published and have no citations at the time of writing.

QiandaoEar22 [122], [128], originally published by Du and Hong in May 2024, is a multi-target acoustic dataset consisting of 9 hours and 28 minutes of signal data and 21 hours and 58 minutes of background noise data.

The recordings were taken over four days between the 24th and 28th of June, 2022. The recording locations were off the coast of Houdao Island in Qiandao Lake, located in China's

Zhejiang Province. According to the authors, the recording period coincided with the local tourism season, causing a variety of vessel classes to be recorded such as speedboats, large tour boats, cargo ships, sanitation vessels, small boats, and research ships [122, p. 5].

The authors used a single DigitalHyd SR-1 hydrophone (the same model used by the ShipsEar team) anchored using a counterweight and buoy system to set its depth. The sampling rate was 52,734 Hz with a sampling resolution of 16 bits. The water depth ranged from 30 to 50 metres and the hydrophone was placed at a depth between 10 and 15 metres.

Due to QiandaoEar22 being a multi-target dataset, the structure of the dataset is more complex. Each file is in the format:

`YYYYMMDDHHMMSS_targetType_name_distance_audibility[&name_distance_audibility]*.wav`

where:

- `YYYYMMDDHHMMSS` is a 14-digit timestamp (year, month, day, hour, minute, second).
- `targetType` is either `S` (single target) or `M` (multiple targets).
- `name` is the concatenated name of the target (alphanumeric characters).
- `distance` is `F` (far), `M` (medium), or `N` (near).
- `audibility` is `S` (strong), `M` (middle), or `W` (weak).
- `[&name_distance_audibility]*` represents optional additional targets, separated by `&`, following the same format.
- `.wav` is the file extension.

The dataset can be downloaded at `https://github.com/xiaoyangdu22/QiandaoEar22`.

QiandaoEar22 offers a much-needed new dataset to the field of underwater acoustics. At a duration of almost nine and a half hours, the dataset sits somewhere in-between ShipsEar and DeepShip in terms of volume. However, the complexity of the multi-target format may present certain challenges. Handling and analysing multiple overlapping sources of noise could complicate data processing and model training. Researchers will need to develop sophisticated methods to separate and classify these signals effectively. It will be interesting to see how the academic and practical applications evolve as researchers utilise this dataset, potentially leading to new insights and advancements in underwater acoustics.

Oceanship

In August 2024, Li et al. introduced a new dataset into the field of underwater acoustics called Oceanship [123]. The dataset comprises a total of 107,540 recordings, covering 15 vessel classes and spanning 121 hours and 10 minutes in duration. The dataset is divided into two subsets: Oceanship CG (coarse-grained), which contains 53,771 recordings, and Oceanship FG (fine-grained), which includes 53,769 recordings. The key distinction

between the two lies in the level of annotation; the fine-grained version provides additional detailed metadata on vessel location, heading, and speed.

Oceanship is essentially a curated compilation of audio recordings sourced from Ocean Networks Canada (`https://www.oceannetworks.ca`) captured between 2021 and 2022. The authors decoded Automatic Identification System (AIS)-encoded data to ensure accurate vessel labeling. By adhering strictly to AIS data processing protocols, the dataset ensures high-quality annotations, adding significant value to its use in UATR tasks.

The scale of Oceanship is particularly noteworthy. The total duration of its recordings is more than twice the length of the DeepShip dataset and over 40 times longer than ShipsEar. This sheer volume of data makes Oceanship the largest publicly available UATR dataset, offering substantial training data for deep learning models.

Another key strength of the dataset is its broad coverage of vessel categories. The 15 classes included in Oceanship account for 83% of vessel names listed in the AIS naming protocol. These categories also overlap significantly with those used in DeepShip and ShipsEar, which raises the possibility of utilising Oceanship as a pre-training resource for models aimed at classifying vessels in these other datasets. Moreover, the detailed metadata available in Oceanship FG, particularly vessel heading and speed information, offers a level of annotation that is absent in DeepShip, providing researchers with richer data for model development and analysis.

Oceanship has the potential to surpass DeepShip as the benchmark dataset for UATR. However, at the time of writing, the dataset has yet to be fully accessible to the public. Issues with the uploaded files, particularly the FG subset, have prevented users from downloading the complete dataset. Once these issues are resolved, Oceanship could become an indispensible resource for the UATR community.

Challenges

Despite the availability of datasets like ShipsEar and DeepShip, many researchers continue to rely on proprietary and often confidential datasets, greatly reducing the reproducibility of their results. There are numerous examples of this; for instance, Shin et al. collected 66 minutes of data from a single ship navigating through the Korean Peninsula but noted that, "because we used passive sonar systems developed for defence purposes, details such as the sampling frequency, sensor array design frequency, sensor distances, and frequency components of the radiation signal cannot be disclosed" [129, p. 8].

This lack of transparency is a pervasive issue in the field of UATR. To assess the extent of this problem, we compiled and reviewed a set of recent papers in the field, categorising them based on whether they utilised proprietary datasets. Out of the 80 papers examined, 54 employed private datasets – an overwhelming majority. This issue is consistently noted across various reviews in the field, with Aslam et al. [4, p. 16], Domingos et al. [3, p. 24], Neupane and Seok [5, p. 24], Luo et al. [6, p. 12], and Niu et al. [1, p. 19] all identifying the lack of data availability as a critical, if not the most pressing challenge facing UATR research. In fact, Domingos et al. goes as far to say that the lack of common datasets

"renders this type of research largely insubstantial, since without a common body of data, comparing and benchmarking solutions is a virtually impossible task" [3, p. 24]. They further remark that in order to facilitate reproducibility, both the code and datasets used in journal papers should be made public – "otherwise, the very purpose of publishing results is defeated" [3, p. 24].

Moreover, even when public datasets are used, inconsistency in how data is divided between training and test sets can significantly skew results. Both DeepShip and ShipsEar serve as valuable benchmarks for research, yet these databases do not enforce a standard method of dividing data, leading to discrepancies in performance metrics across studies. For instance, some researchers divide audio records into multiple segments and assign these segments randomly to training and test sets, which can lead to the same underlying recording being represented in both sets. Since ship-radiated noise tends to remain stable over time, this can cause models to overfit, inflating their performance. To prevent such information leakage, it is recommended that entire audio records be used for either training or testing, but not both. Standardising such data division methods would allow for more rigorous validation and enable more meaningful comparisons across studies [1].

Chapter 3

Establishing a Baseline Performance

The objective of this thesis is to enhance the classification of underwater acoustic signals by refining preprocessing techniques and feature extraction methods. As with any scientific endeavour, it is essential to first establish a benchmark – a reference point that serves as the foundation for measuring progress. This chapter lays out the baseline classifier architecture that will be used consistently throughout all subsequent experiments. By keeping the classifier architecture constant, we ensure that any performance improvements are solely attributed to changes in the preprocessing or feature extraction methods, rather than from adjustments in the classifier.

This chapter is organised as follows. We begin by introducing the dataset used throughout this thesis, including an overview of its composition, our data processing approach (segmentation, fold generation etc.), and a discussion of standardised splitting techniques found in the literature. Next, we focus on our chosen input features – power spectrograms. We detail the feature extraction pipeline, including preprocessing steps such as downsampling, normalisation, spectrogram generation parameters, frequency cutoffs, and resizing. We then present the classifier architecture – a CNN-LSTM model chosen for its ability to extract both spatial and temporal features. This section includes a detailed architecture layout, training configuration, hyperparameter tuning strategies, as well as implementation details in TensorFlow Keras. Challenges encountered in implementation due to dataset size are also discussed. Finally, we provide the benchmark results of the CNN-LSTM model trained with the baseline feature extraction pipeline. These results serve as a reference point for evaluating techniques introduced in subsequent chapters.

3.1 The dataset: DeepShip

The dataset selected for this study is DeepShip [117]. There are two key reasons behind this choice. Firstly, DeepShip has become the authoritative benchmark dataset for UATR, with many recent scientific studies reporting their performance metrics on this dataset [130]–[142]. Given its extensive use, the dataset provides a solid foundation for evaluating

new techniques, allowing direct comparison with a vast body of prior work. Additionally, past research students at Thales have also worked with DeepShip which provides continuity for our work and the opportunity to build upon their results [143].

Secondly, while the newly released OceanShip dataset (see Section 2.3.4) offers more recordings than DeepShip – over twice the length in minutes – it is currently inaccessible to those outside of mainland China. Even supposing that the dataset was available, the class labels within OceanShip have not been externally validated, raising concerns over the quality and reliability of its data. In contrast, DeepShip has undergone independent verification by sonar processors at Thales, ensuring its labels are accurate and reliable for research purposes.

3.1.1 Dataset overview

As outlined in Section 2.3.4, the DeepShip dataset consists of 613 recordings from 265 ships, categorised into four vessel classes (see Table 2.7 for more information). However, some of the recordings in the original dataset lack metadata, leading us to work with a refined subset of DeepShip for this thesis. This subset includes 582 of the 613 recordings, representing 249 of the 265 vessels. Consequently, the dataset length is slightly reduced to 2684 minutes, compared to the original 2824 minutes. Table 3.1 compares the number of ships, recordings, and total recording time for each vessel class in our subset (values outside parentheses) with the original dataset (values inside parentheses). Hereafter, this subset will be simply be referred to as DeepShip.

Table 3.1: Comparison of the DeepShip subset used in this thesis (values outside parentheses) with the complete DeepShip dataset (values in parentheses).

Vessel class	Ships	Recordings	Total recording time (min)
Cargo	62 (69)	106 (110)	628 (640)
Passengership	43 (46)	180 (193)	715 (742)
Tanker	128 (133)	234 (240)	735 (765)
Tug	16 (17)	62 (70)	606 (677)
Total	**249 (265)**	**582 (613)**	**2684 (2824)**

Like many real-world datasets, DeepShip exhibits a minor class imbalance, with some ship classes being more heavily represented than others. A class distribution analysis was conducted to visualise this imbalance. Figure 3.1 illustrates the variation in the duration of recordings for a vessel. As shown, there is at least one outlier vessel in each class that is over-represented. For example, the median recording duration of all cargo vessels is 5.9 minutes, yet the cargo vessel `PRINCESS_SUPERIOR` has over 151 minutes of sonar signal data. Also, the number of ships and number of recordings in each class is quite unbalanced (Figure 3.2), however the overall total recording time of each class is fairly balanced (Table 3.1).

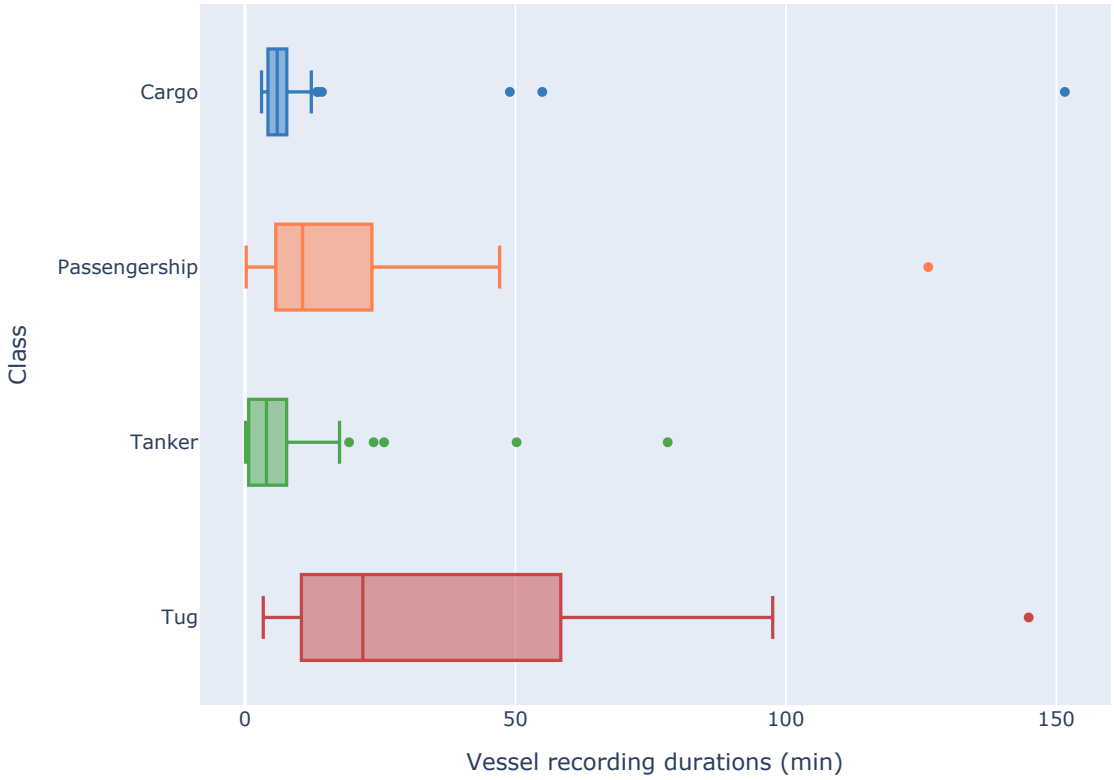
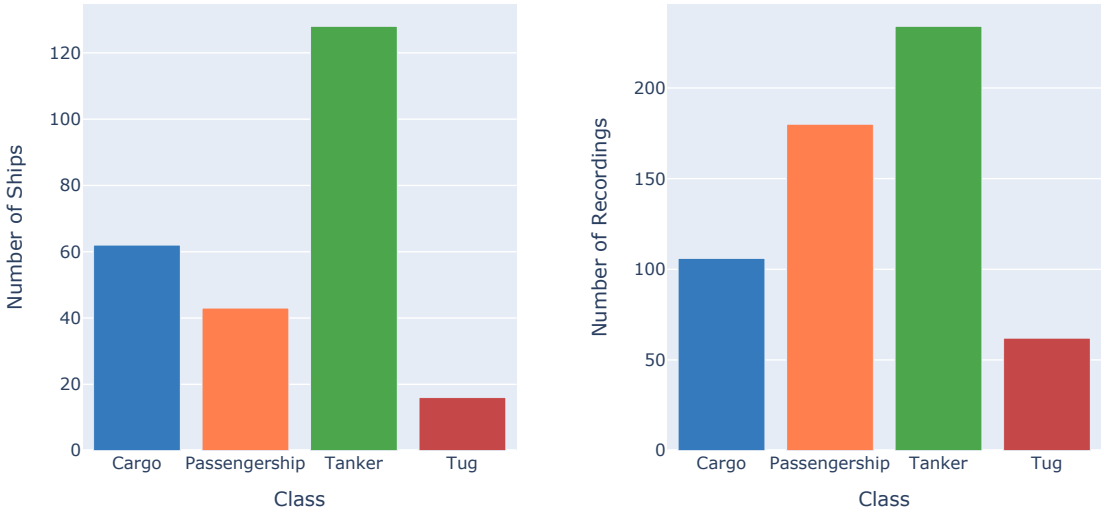


Figure 3.1: The distribution of vessel recording durations in the DeepShip subset, split by class.



(a) Ships **(b)** Recordings

Figure 3.2: The number of (a) ships and (b) recordings in each class of the DeepShip subset.

3.1.2 Recording segmentation

A widely used approach in audio applications for machine learning is to split each audio recording into smaller segments to boost the number of training samples, improve granularity in feature extraction, and allow more robust classification models to be trained. Segmenting the data into smaller clips enables better generalisation by providing the model with more diverse examples and variations across the data. Additionally, smaller, more homogeneous segments can allow the model to focus on the most relevant features while ignoring background noise components, thereby improving classification performance [139]. Some examples of UATR literature which employ segmentation on the DeepShip dataset include: Li et al. [142], who segmented the DeepShip dataset into 5-second clips to yield 11,174 samples, using a 70-20-10 split for training, verification, and test datasets respectively; Chen et al. [136], who segmented DeepShip into 5-second segments and then divided each class into five equal folds at random; and Sun and Leo [141], who used a 3-second segmentation strategy for DeepShip, acknowledging the importance of segmenting audio to increase the number of samples and make the dataset more computationally feasible for machine learning models.

The choice of 3-second segments, specifically, is also grounded in previous studies, which have demonstrated that this interval strikes an optimal balance between maintaining sufficient audio context for accurate classification and managing the model's computational complexity. Longer segments may provide more stability and context, however they also increase computational demands and the risk of overfitting by capturing more irrelevant or noisy information [135], [142]. Conversely, shorter segments may lack sufficient context, with burst noise or transient phenomena in the signal having a more pronounced negative impact on classification accuracy [134], [144]. Thus, the 3-second segment length aims to achieve an effective balance between data complexity and computational efficiency. This segment length is consistent with other state-of-the-art studies that similarly found 3-second segments optimal for underwater acoustic target recognition tasks [133], [134]. Notably, the original authors of the DeepShip dataset also chose to divide each recording into 3-second segments, further validating this selection [117].

In total, DeepShip, with a total recording time of 2684 minutes (161,040 seconds), was segmented into 53,680 3-second segments. After removing segments containing only background noise or silence – to ensure that the inputs were actually representative of ship noise patterns – there were a total of 53,503 valid segments remaining.

3.1.3 Distribution of segments into folds

In auditory applications of machine learning, it is common practice to divide input audio clips into k folds for cross-validation [135], [136], [145]–[147]. The most common approach is *leave-p-out cross-validation*, where p folds are used for training and the remaining $k - p$ folds for testing. This process is repeated until each fold has been used for both training and testing. In cases where the dataset allows, *stratified k-fold cross-validation* is preferred, as it ensures that each fold contains an equal number of samples from each class. However,

this ideal design is not always feasible due to the original dataset's class distribution.

Hence, following best practice, after segmentation the dataset was divided into 10 folds with an effort to maintain an approximately equal number of 3-second segments from all four ship classes in each fold. The complete distribution of samples across the 10 folds is shown in Figure 3.3.

Two key considerations guided the design of the 10-fold split for DeepShip. First, to prevent the model from learning recording-specific characteristics, such as background noise or vessel-specific nuances, all clips derived from a given recording were placed in the same fold [145]. Secondly, given that UATR datasets often contain multiple recordings from the same vessels, we aimed to distribute an approximately equal number of samples from each vessel across the folds. This not only helps balance the class distribution within each fold but also ensures that each vessel is consistently represented across the folds, improving the robustness of the model evaluation.

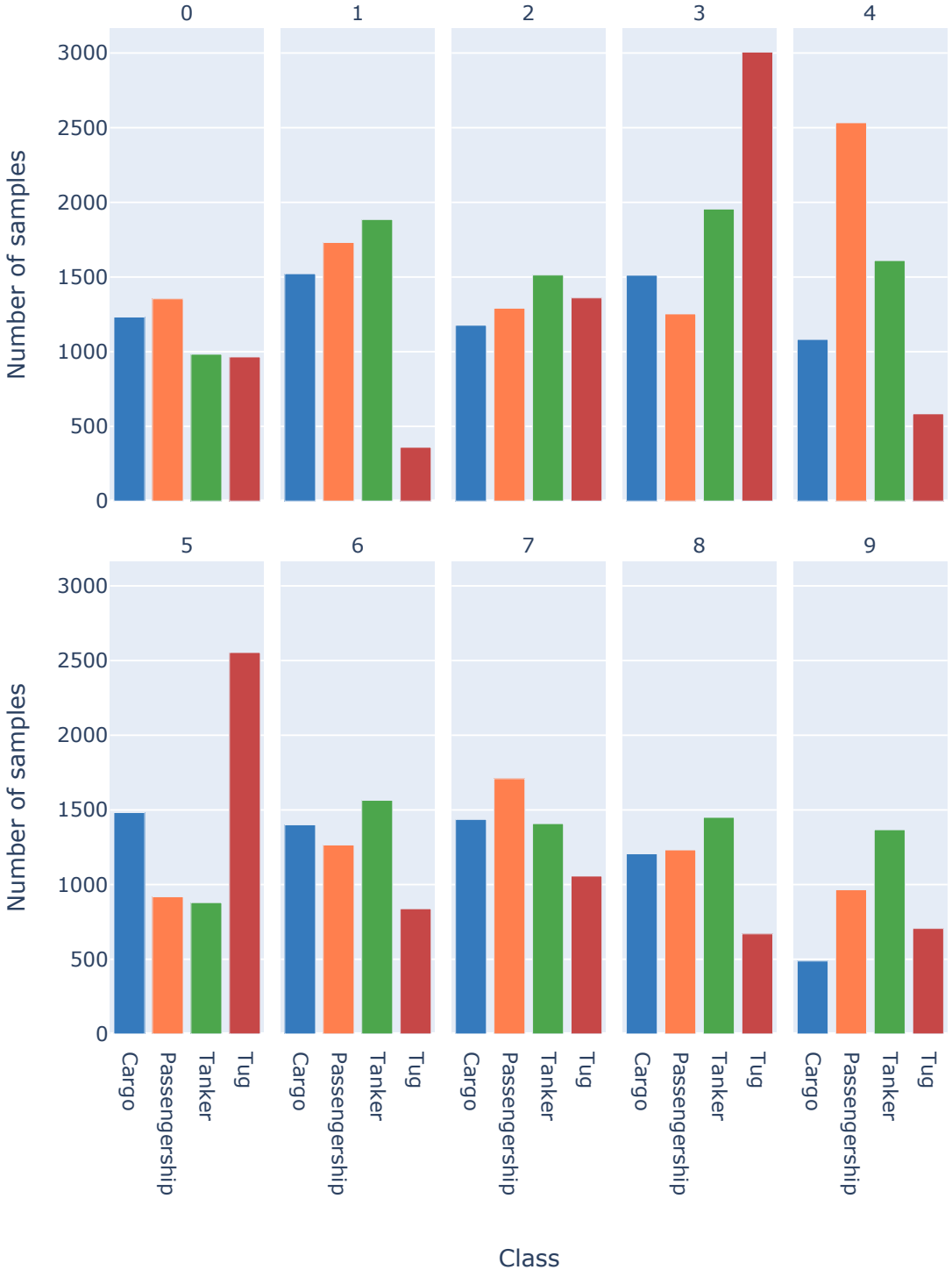
It should be noted, however, that the split is not perfectly uniform; some variability exists in the number of segments per fold (Figure 3.3c). For instance, the Tug class exhibits the largest variation, with Fold 1 containing only 360 segments, while Fold 3 contains 3006. This variability arises from the constraints mentioned above: entire recordings, rather than individual segments, were assigned to each fold to avoid splitting a recording across multiple folds. Furthermore, efforts to distribute an equal number of samples from each vessel across the folds were limited by the inherent imbalance in recording durations within the DeepShip dataset (Figure 3.1). Despite the resulting variability, this method still enhances the dataset's integrity by ensuring that no recording is split across multiple folds, ultimately improving the consistency and robustness of the dataset for cross-validation.

3.1.4 Other standardised approaches

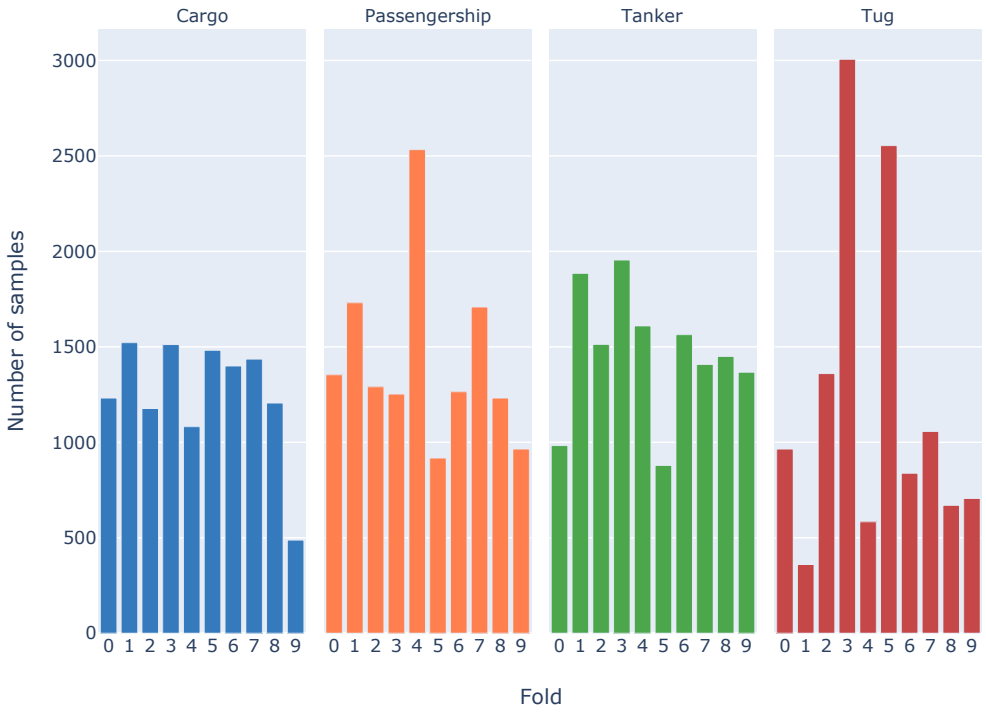
Given the significant variation in segmentation and splitting methodologies highlighted by Niu et al. [1] and previously discussed in Section 2.3.4, there has been a recent trend towards adopting standardised train-test splits to enable more direct comparisons between classification techniques. In 2023, Zhu et al. proposed a specific division of the DeepShip dataset into 3-second segments for a consistent training and testing split [148]. This standardised split was published online [149], and several subsequent studies have since adopted this methodology to ensure comparability across research [133], [139], [150], [151].

Zhu et al. also introduced a standardised background noise class to complement the DeepShip dataset. Unlike other datasets such as QiandaoEar22, DeepShip lacks a dedicated background noise class due to the high level of shipping activity in the recording area, which means all recordings contain some degree of ship noise [117]. Although the original authors of DeepShip used over seven hours of external background noise recordings in their classifier, they did not disclose or publish these files. To address this, Zhu et al. provided a set of background noise files for future studies.

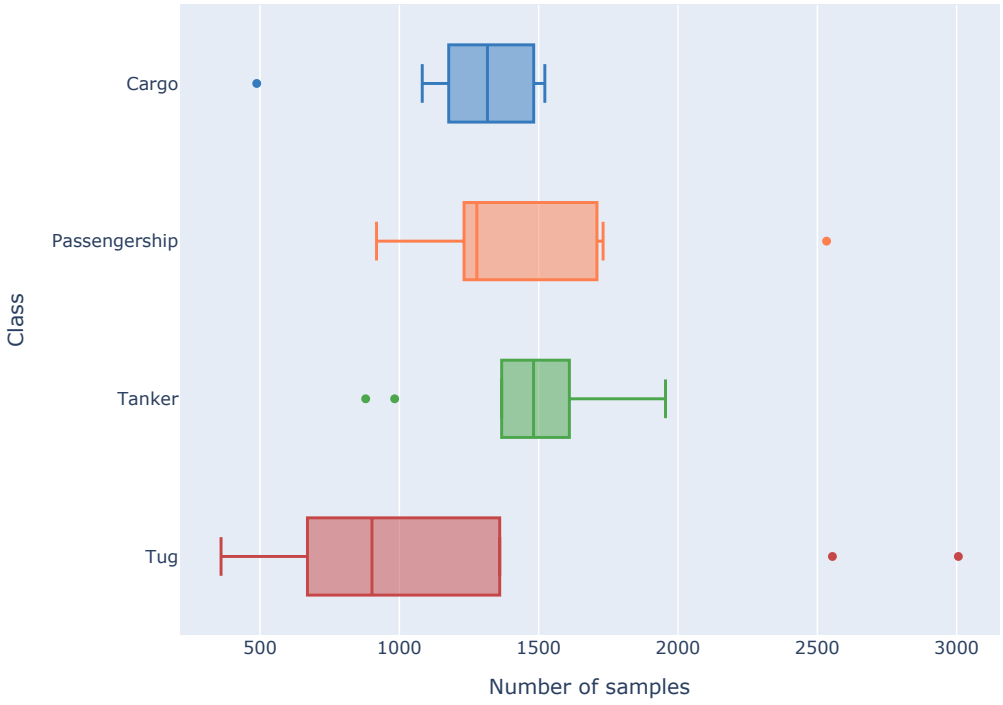
However, the inclusion of external background noise remains a point of debate. Some researchers, such as Tian et al., have incorporated background noise to improve classifi-



(a) Count of samples in each fold by class.



(b) Count of samples in each class by fold.



(c) Spread of the number of samples across folds by class.

Figure 3.3: Distribution of samples across 10-fold split.

cation performance [140]. However, others argue that this approach may not accurately reflect the specific acoustic environment of the DeepShip recordings.

For this study, we adhered to our own splitting methodology, consistent with the practices of Thales researchers, to maintain continuity across our internal projects. Furthermore, we opted not to augment the dataset with random noise samples, as we do not consider this step necessary or appropriate for accurately reflecting the operational environment represented in DeepShip.

3.2 Input features

The input feature chosen for our benchmark model was the power spectrogram, a transformed version of the conventional spectrogram designed to enhance interpretability and align with human auditory perception. Power spectrograms apply a logarithmic transformation to the conventional spectrogram, calculated as $10 \cdot \log_{10}(P + \epsilon)$, where P is the power or magnitude of the signal and ϵ is a small constant to avoid zero values. This logarithmic scaling brings the power spectrogram more in line with human auditory perception and also emphasises subtler sound components, making it both more visually interpretable and informative for analysis. The decision to use power spectrograms is well-supported by existing literature in underwater acoustics. For instance, previous studies, such as Cao et al. [152], have demonstrated the use of power spectrograms for extracting unique frequency features associated with vessel machinery, such as shaft frequencies.

3.2.1 Processing pipeline

Downsampling Each audio recording was downsampled from the original sampling rate of 32 kHz to 5 kHz. This choice was guided by studies such as Arveson and Vendittis [124], Malinowski et al. [31], Pricop et al. [125], and McKenna et al. [127], which suggest that the majority of discriminative acoustic information for modern commercial vessels lies within the 0-2 kHz frequency range. Hence, in-line with the Nyquist-Shannon sampling theorem, 5 kHz was chosen to maintain the most important frequency features whilst also reducing computational complexity.

Audio normalisation Each audio clip then underwent 0-1 normalisation to standardise amplitude levels across samples. This step aimed to mitigate the effects of loudness variations, allowing the machine learning model to focus on frequency-based characteristics rather than amplitude fluctuations.

Spectrogram generation The spectrograms were computed using MATLAB's `spectrogram` function, with parameters fine-tuned over several iterations to achieve an optimal balance between time and frequency resolution. Table 3.2 lists the final parameters used for spectrogram generation. The window length of 40 milliseconds was selected to achieve a balance between frequency resolution (for capturing tonal components) and temporal

resolution (for capturing transients). A 75% overlap was applied between consecutive windows to minimise information loss, and the FFT length was set to 1024 points, providing sufficient spectral detail. The output of the `spectrogram` function is the amplitude spectrogram S, the short-time Fourier transform of the input signal returned as a matrix. This output was subsequently converted into its power representation using the transform $P = 10 \cdot \log_{10}(|S|^2 + 10^{-8})$.

Table 3.2: Final short-time Fourier transform parameters for power spectrogram computation.

Parameter	Value
Sampling frequency (`fs`)	5 kHz
Window length (`window`)	40 ms = 200 segments, rounded to 256
Overlap (`noverlap`)	75% of window length
Fast Fourier transform length (`nfft`)	1024 points
Output dimensions	195×231 (frequency bins $\times$ time bins)

Amplitude cutoff To suppress low-intensity regions dominated by background noise, the spectrogram amplitudes were limited to a minimum of -30 dB. This threshold was chosen based on the analysis of amplitude distributions across all vessel classes. Figure 3.4 illustrates the average amplitude histograms for the four vessel classes, highlighting that the majority of amplitude values are concentrated between -30 dB and 20 dB. Cargo and Tanker classes exhibit the widest amplitude ranges, extending from approximately -40 dB to 30 dB, while Passengership and Tug classes display narrower ranges with fewer extreme values. Most importantly, amplitude values below -30 dB are uncommon, justifying the cutoff.

The main benefit of applying this threshold is that it removes the influence of an extremely small number of low-amplitude values (< -30 dB) on the color scaling of the spectrograms. This ensures a more uniform color mapping. Figure 3.5 compares spectrograms from the same recording with the same STFT parameters before and after applying the low amplitude cutoff. In the unprocessed spectrogram, the presence of low-intensity noise has a "blurring" effect on the figure, while the spectrogram with the cutoff highlights narrowband events with much greater sharpness and clarity.

Frequency cutoffs To optimise SNR and improve computational efficiency, two frequency-based preprocessing steps were applied to the power spectrogram matrices. First, a low-frequency cutoff was introduced to exclude components below 50 Hz, which are often associated with DC offset and other artifacts which typically contain noise and large amplitude variations that do not significantly contribute to vessel classification. Removing them helps the power spectrogram better emphasise the informative portions of the spectrum, thereby enhancing SNR.

Second, a high-frequency cutoff was applied at 1000 Hz based on an analysis of the

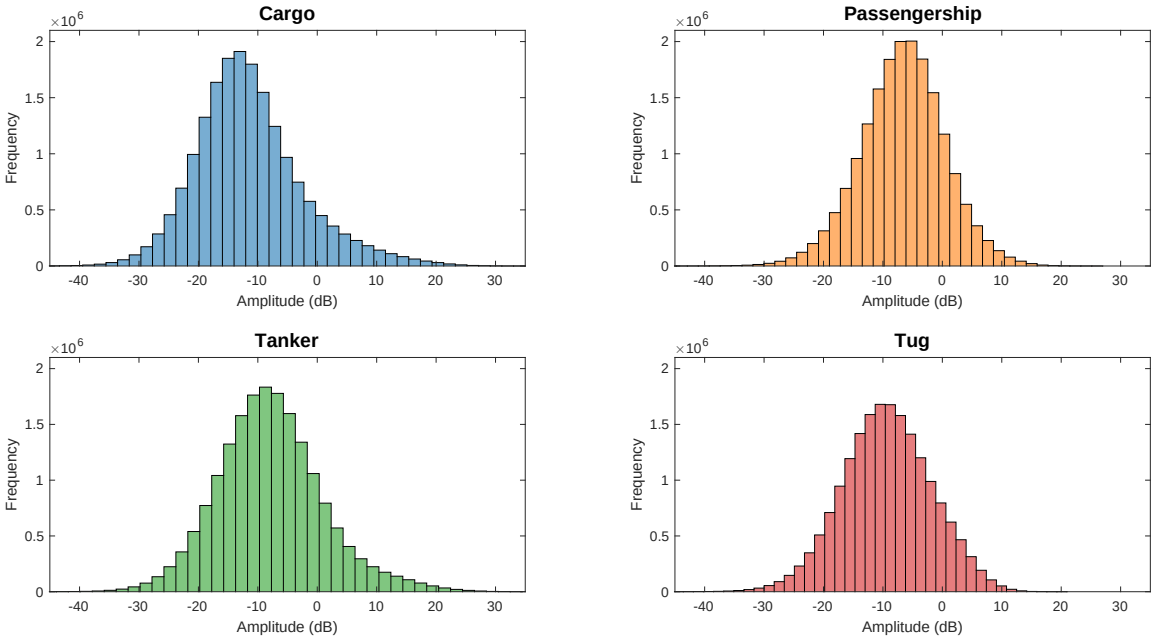


Figure 3.4: Average amplitude histograms for each vessel class.

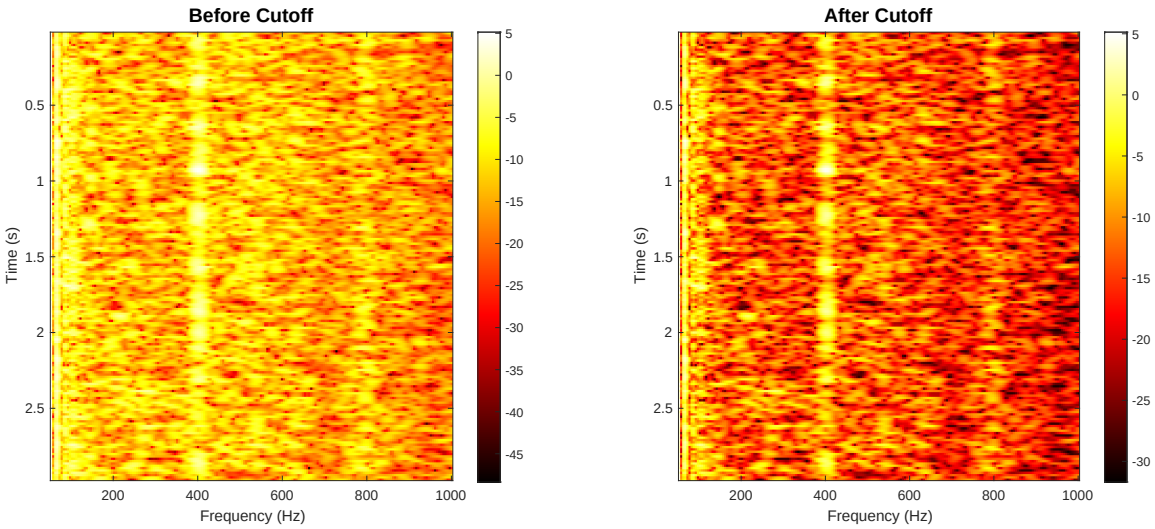


Figure 3.5: Spectrograms before and after amplitude cutoff. The cutoff at −30 dB suppresses low-intensity noise, enhancing visual clarity.

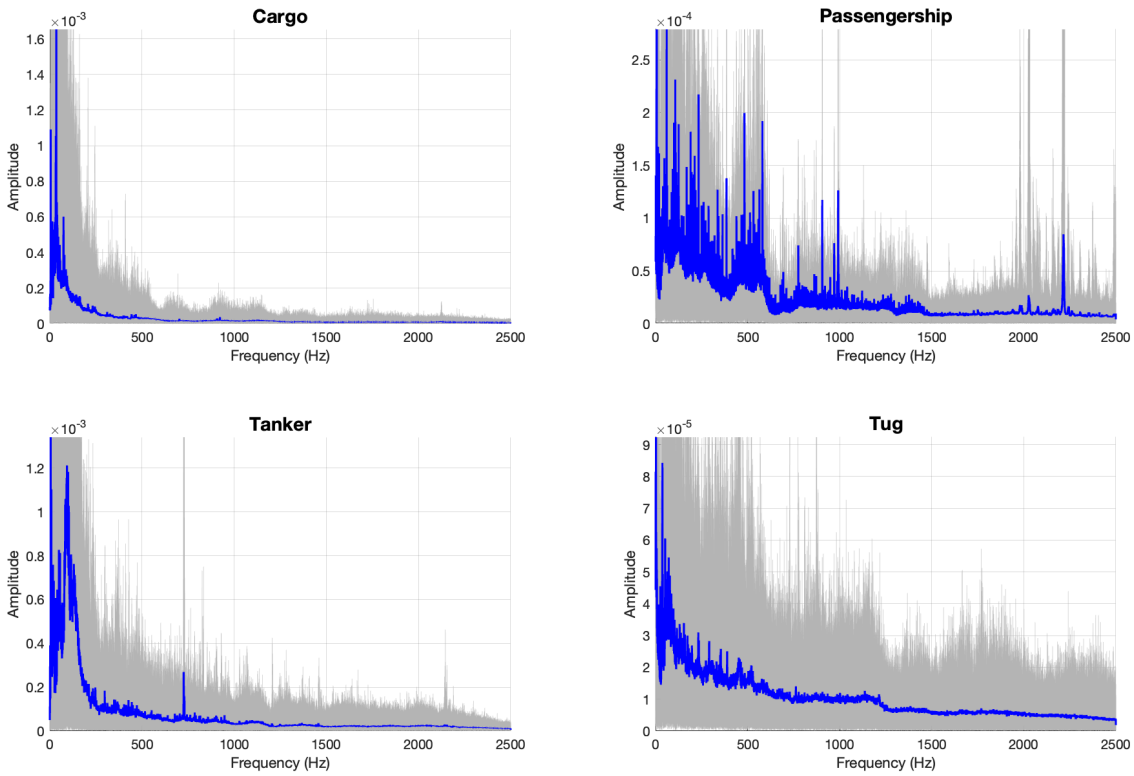


Figure 3.6: Average single-sided amplitude spectra of each class in the DeepShip dataset. Faint gray lines represent individual segments, while the bold blue line indicates the average amplitude spectrum for each vessel type averaged over 100 segments.

frequency spectra for the vessel types in the DeepShip dataset. As shown in Figure 3.6, significant energy is concentrated below 500 Hz for all classes with a marked decline in amplitude beyond 1000 Hz. By the range of 1500-2000 Hz, the amplitude is nearly negligible, indicating minimal useful information at these higher frequencies. Limiting the spectrogram to 1000 Hz retains the most relevant frequency content while reducing dimensionality and potential noise, as supported by [153].

Resizing The original spectrogram output had dimensions of 195×231. In order to ensure compatibility with convolutional layers used in our model, these were resized to 192×192 using MATLAB's `imresize` function. Examples of the final power spectrograms which were used as input to the machine learning model are shown in Figure 3.7.

Exporting To facilitate efficient storage and loading of spectrograms during the model training phase, the computed power spectrograms were exported as `.mat` files. This choice was made due to the space efficiency of the `.mat` format compared to CSV files as well as its parsing efficiency with Python via the `scipy.io.loadmat` function. It achieves this because `.mat` files store the numerical arrays as structured objects, rather than CSV files which are text-based. This makes it ideal for handling large datasets of spectrogram matrices without excessive file I/O overhead.

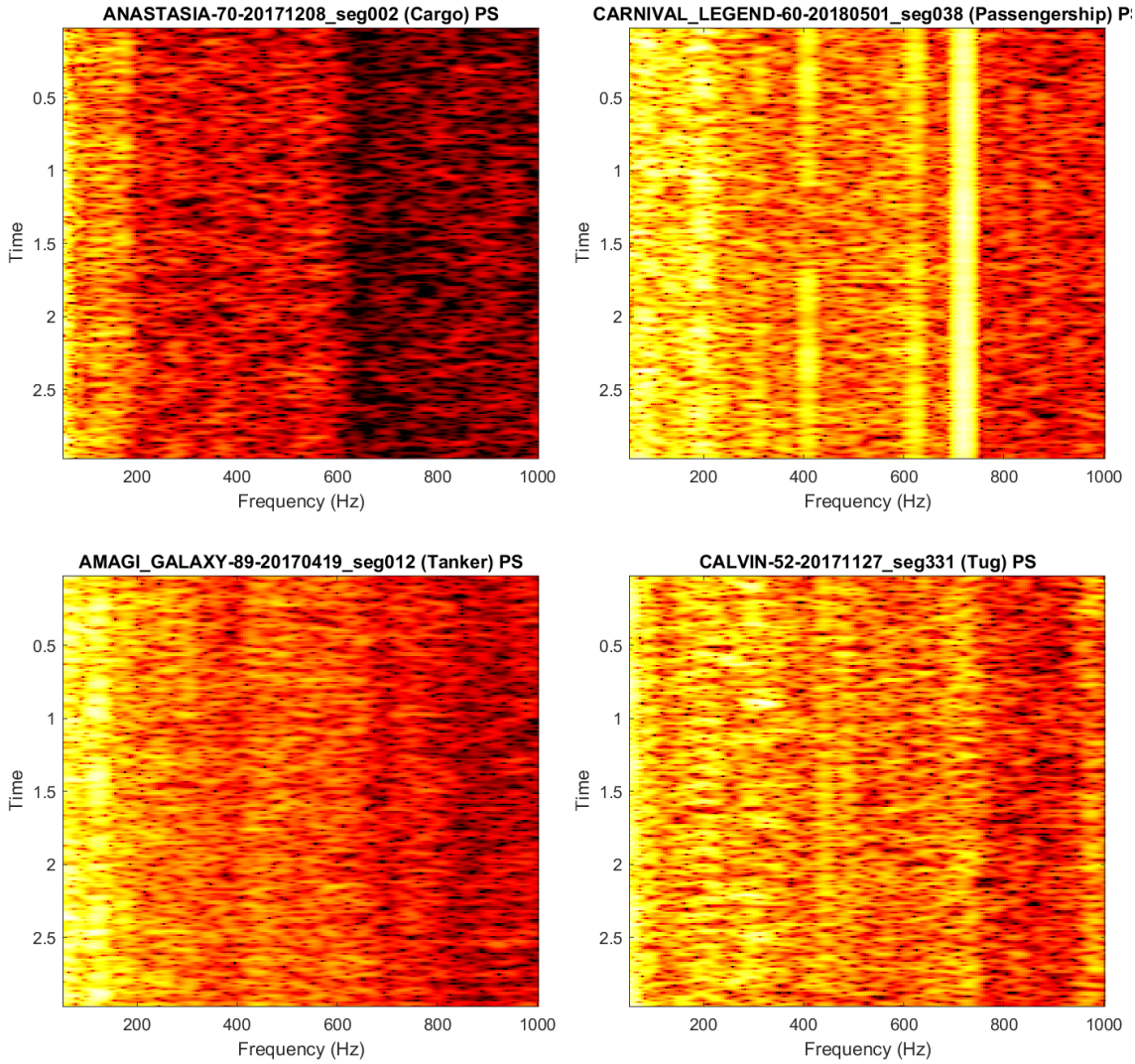


Figure 3.7: Examples of power spectrogram representations for each vessel class in the DeepShip dataset.

3.2.2 Implementation

The generation of power spectrograms was implemented using MATLAB in a way that ensured flexibility, reproducibility, and clarity in the preprocessing steps. At the core of this pipeline is our custom MATLAB function `wavToSpec()`, which accommodates all the preprocessing steps previously mentioned and allows for them to be enabled or adjusted through simple parameter toggles. This modular design not only simplifies experimentation and debugging but also ensures consistency across different stages of the thesis. Additionally, the pipeline serves as an effective documentation tool, enabling future researchers to easily understand and replicate the spectrogram generation process. All MATLAB code was written and tested in version 2024a, and is available on the project's GitHub page [154].

3.3 The classifier: CNN-LSTM

The chosen classifier for the baseline model is a hybrid convolutional neural network–long short-term memory (CNN-LSTM) model, which aims to leverage the strengths of both CNNs for spatial feature extraction and LSTM networks for sequential data processing. CNNs excel at analysing spatial data, such as images or spectrograms, while LSTMs are effective at modeling time-series data, identifying temporal patterns across sequences. By integrating CNN and LSTM, the hybrid architecture can capture both spatial and temporal features, making it well-suited for tasks such as time-series forecasting [155], [156], natural language processing [157], [158], and, as in our case, image classification in complex domains [159], [160].

In the context of UATR, this architecture serves as an ideal benchmark due to the spatial and temporal nature of sonar signal data. CNNs can extract meaningful spatial features from spectrograms, while LSTMs sequentially process these extracted features, capturing unique temporal patterns that vary across vessel types due to differences in machinery, propulsion, and operating speeds. Thus, we believe that the CNN-LSTM classifier is well-positioned to benchmark the DeepShip dataset for this thesis.

3.3.1 Architecture layout

The architecture starts with four convolutional blocks, each contributing to progressively deeper feature extraction, followed by an LSTM layer for temporal pattern recognition and concluding with dense layers for final classification. A summary of each component is given below, and a diagram of the network is presented in Figure 3.8.

Convolutional blocks 1 and 2 The first convolutional block applies a 2D convolution layer with 128 filters and a 5×5 kernel size, using padding to maintain spatial dimensions. The convolutional layer is followed by batch normalisation, which accelerates training and stabilises the learning process. A ReLU activation function introduces non-linearity, and a 2×2 max-pooling layer with a stride of $(2, 2)$ down-samples the feature map, reducing

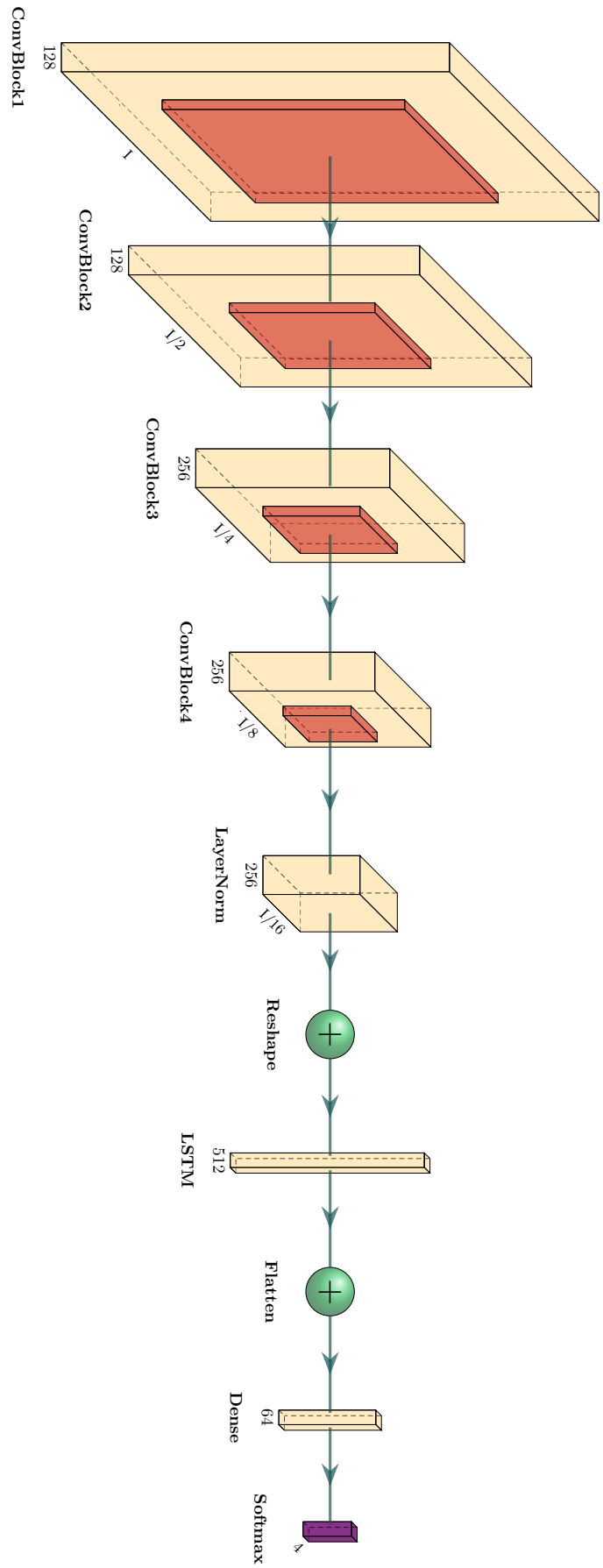


Figure 3.8: Baseline CNN-LSTM architecture diagram, illustrating key layers and their configurations. Each ConvBlock includes batch normalisation and ReLU activation, though these are not explicitly shown. Created using [161].

spatial dimensions while retaining essential features. The second convolutional block is identical to the first, however, here the max-pooling size is increased to $(4, 2)$ to further condense the feature representation.

Convolutional blocks 3 and 4 The third and fourth convolutional blocks replicate the structure of the first two blocks but with double the number of filters to enable the capture of more complex feature hierarchies. Both blocks use a 5×5 kernel, batch normalisation, ReLU activation, and max-pooling layers; however, in the final block, the max-pooling layer again uses a larger pooling window $(4, 2)$ to further down-sample the feature map, preparing it for sequential processing.

LSTM layer After the convolutional blocks, the model incorporates a layer normalisation step to standardise the data. The output is then reshaped before being passed into the LSTM layer with 512 hidden units. The LSTM layer uses the hyperbolic tangent (tanh) as its activation function. This choice is based on TensorFlow's optimisation capabilities; when tanh is used as the activation function, TensorFlow can use an efficient cuDNN implementation on compatible GPUs, accelerating LSTM computations and maximising performance.

Classification layers Following the LSTM layer, the output is flattened and passed through two fully connected (Dense) layers. The first dense layer, with 64 units and a ReLU activation, reduces the dimensionality and enables more complex learned representations, while the final dense layer, with 4 units and a softmax activation, outputs the class probabilities for each of the four vessel types in the DeepShip dataset.

3.3.2 Training configuration

For training, we used a GPU batch size of 16, the Adam optimiser [162], and categorical cross-entropy as the loss function. We set the learning rate to 1×10^{-5}.

As previously explained, to ensure a robust evaluation of the classifier's performance, we employed a 10-fold leave-two-out cross-validation approach for training. Practically, this meant that each iteration, eight folds were used for training, one fold was used for validation, and one fold was used for testing. This process was repeated 10 times, with each fold serving as the testing or validation set exactly once. The accuracy and loss were averaged across all 10 folds to provide a reliable estimate of the model's performance. This method helps mitigate the potential for overfitting to any specific subset of the data and provides a more comprehensive measure of the classifier's generalisation ability. A summary of the final training parameters can be found in Table 3.3.

All training was conducted on a Windows machine equipped with a NVIDIA GeForce RTX 2080 GPU (8GB) and 64GB of RAM. The baseline model was implemented using TensorFlow Keras (v2.10.0), and trained with CUDA (version 12.6), cuDNN (version 8.1.0),

and cudatoolkit (version 11.2) libraries which are necessary for accelerated deep learning on NVIDIA GPUs.

Table 3.3: Final training parameters for benchmark CNN-LSTM model.

Parameter	**Final value**
GPU batch size	16
Optimiser	Adam
Loss function	Categorical cross-entropy
Learning rate	1×10^{-5}
Validation approach	Leave-two-out 10-fold cross validation
Evaluation metrics	Accuracy, F1-score

3.3.3 Hyperparameter tuning

The final architecture and training configuration described in the preceding sections were determined through an iterative process of hyperparameter tuning. Initially, the model architecture was designed with parameters based on standard practices for CNNs and LSTM layers for time-series and spectral data. Specifically, the CNN layers used a filter size of 3×3 with 64 filters, while the LSTM layer included 1024 hidden units. The learning rate for training was set at 1×10^{-3}.

We then employed the Keras Tuner library to try and enhance our architecture by performing hyperparameter tuning. We used Keras Tuner's `RandomSearch` method, which is particularly well-suited for projects with large search spaces and limited computation resources, as alternative methods (such as Bayesian optimisation) often require the direct modelling of dependencies between parameters – a computationally expensive step. Using `RandomSearch` allowed us to find the most promising configurations in a relatively short time with low computational overhead.

The hyperparameters included in the search space, as well as a brief motivation for tuning each of them, are provided below.

- Number of filters and filter size: We tested kernel sizes of length $[3, 5, 7]$ to evaluate how receptive field size affects the model's ability to capture spatial features. We also varied the number of filters in each convolutional layer across $[32, 64, 128, 256]$.
- LSTM units: We tested LSTM configurations with $[256, 512, 1024, 2048]$ units to determine the optimal memory capacity for capturing temporal dependencies.
- Dense layer units and activation function: For the dense layer immediately after LSTM, we experimented with $[32, 64, 128, 256]$ units and evaluated both `relu` and `tanh` activations to observe which non-linearity best captures high-level abstractions in the data.
- Optimiser: The choice of optimiser can significantly impact convergence; we compared `adam`, `rmsprop`, and `sgd`.

- Learning rate: Four different learning rates were tested to find the best balance between learning speed and stability: 10^{-5}, 10^{-4}, 10^{-3} and 10^{-2}.

A summary of the search space alongside the hyperparameter configuration yielding the best performance is provided in Table 3.4.

Table 3.4: Hyperparameter search space used for model tuning with ideal configuration.

Hyperparameter	**Values tested**	**Optimal value**
Filter size (`filter_size`)	3, 5, 7	5
Number of filters (`num_filters`)	32, 64, 128, 256	128
LSTM units (`lstm_units`)	256, 512, 1024, 2048	512
Dense layer units (`dense_units`)	32, 64, 128, 256	64
Dense layer activation (`dense_activation`)	`relu`, `tanh`	`relu`
Optimiser (`optimizer`)	`adam`, `rmsprop`, `sgd`	`adam`
Learning rate (`learning_rate`)	1e-5, 1e-4, 1e-3, 1e-2	1×10^{-5}

The results of the hyperparameter tuning highlight that a larger filter size and an increased number of filters in the CNN layers improved the model's feature extraction capability, while a reduction in LSTM units (from the initial 1024 to 512) helped balance memory capacity with computational efficiency. Additionally, a low learning rate of 1×10^{-5} with the Adam optimiser yielded the best results.

Based on these results, the final network architecture was modified to incorporate the optimal hyperparameters. The CNN was configured with a 5×5 filter size and 128 filters, while the LSTM layer used 512 hidden units. The dense layer was set to 64 units with `relu` activation. Training was carried out with the Adam optimiser and a learning rate of 1×10^{-5}. This process of hyperparameter tuning not only improved the model's performance but also provided insights into the configuration choices that best capture the temporal and spectral features for the DeepShip dataset.

3.3.4 Implementation

The implementation of the CNN-LSTM classifier and training pipeline is built as a reusable and flexible Keras-based codebase. Key features of this implementation are its adaptability to large datasets as well as its ability to perform k-fold cross-validation.

Given the dataset's size of over 53,000 segments, making up over 25GB of data, a critical bottleneck in the DeepShip machine learning pipeline was the handling of spectrogram data. The primary challenge lay in the process of loading spectrogram data from files into memory, which is computationally expensive and prone to causing memory issues if not optimised. To address this, we experimented with several approaches: initially loading all spectrograms into memory, followed by experimenting with TensorFlow's `tf.Dataset` API, and ultimately implementing custom data generators. These generators employed lazy loading – dynamically loading and preprocessing spectrograms in small batches just

before they were fed into the model. By introducing a `DATA_BATCH_SIZE` parameter, we controlled how many spectrograms were read from disk at once, significantly reducing memory usage and enabling stable training. This approach not only avoided memory bottlenecks but also provided flexibility to scale the pipeline for larger datasets or limited hardware configurations.

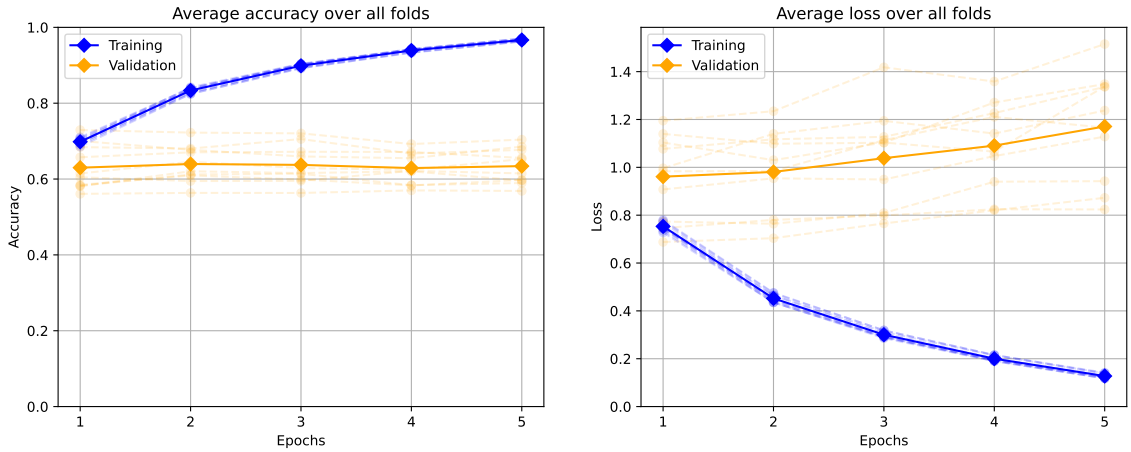
Ultimately, these custom data generators became the foundation of our DeepShip machine learning pipeline, as they enabled our implementation of k-fold cross-validation. The `k_fold_cross_validation` function manages the partitioning of the dataset and the iterative training, validation, and testing process. For each fold, the dataset is divided such that the current fold serves as the test set, the next fold (or the first fold if at the last iteration) is used for validation, and the remaining folds are used for training. This ensures that each fold is used for testing exactly once. The model's performance is evaluated using metrics such as loss, accuracy, precision, recall, and F1-score. These metrics are calculated for each fold and averaged to summarise the model's overall performance. Additionally, the model's weights are reset each fold, ensuring a robust and thorough assessment and providing a reliable benchmark for performance evaluation.

3.4 Baseline results

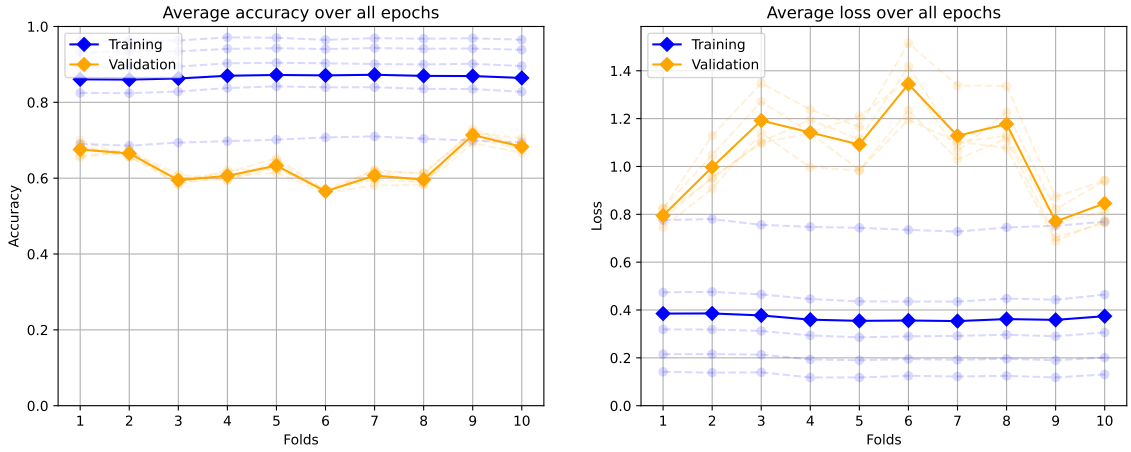
After training the benchmark CNN-LSTM model with the final parameters detailed in Table 3.3 for 5 epochs per fold, we achieved an **overall baseline accuracy of 63.41%**. The model also demonstrated an average precision of 66.53%, recall of 63.41%, and F1 score of 63.75%.

The training and validation accuracy-loss curves, shown in Figure 3.9, provide important insights into the model's performance across the 10-fold cross-validation process. The curves for the benchmark model demonstrate strong consistency and reliability. In Figure 3.9a, the validation accuracy and loss for each fold closely align with the average trend line across all folds, with minimal deviation. This indicates that the model is not only converging effectively but also generalising well across the dataset. Similarly, in Figure 3.9b, the validation accuracy curves for each fold over all epochs are almost identical, and the validation loss curves display remarkable uniformity. The consistency of these curves highlights the robustness of the benchmark model and its ability to learn effectively from the dataset without overfitting or instability.

In conclusion, the benchmark CNN-LSTM model demonstrates a consistent performance, providing a reliable baseline for future improvements.



(a) Accuracy and loss curves across all folds by epoch.



(b) Accuracy and loss curves across all epochs by fold.

Figure 3.9: Training and validation accuracy and loss curves for the benchmark CNN-LSTM model.

Chapter 4

Normalisation

Normalisation is considered a fundamental preprocessing step when using spectrograms for machine learning tasks. Its primary purpose is to standardise input data in order to reduce variability in amplitude scales and improve model training stability. By ensuring that all inputs share similar statistical properties, normalisation enhances convergence and reduces the risk of numerical instability. This chapter aims to quantify the benefits of normalisation for use with underwater acoustic datasets such as DeepShip.

4.1 Introduction

The primary motivation for normalisation is to standardise the input data to avoid biases caused by differences in feature magnitudes. Take the following example as illustration: consider two input features x_1 and x_2 with ranges $x_1 \in [0, 1000]$ and $x_2 \in [0, 1]$ and corresponding weights w_1 and w_2, $w_i \in [-1, 1]$. Now, the activation function at the first layer in the neural network will receive input $\boldsymbol{w}^T\boldsymbol{x} = w_1x_1 + w_2x_2$. Without normalisation, x_1 would dominate the model's computations, making it difficult for the network to effectively learn from x_2. Normalisation mitigates this by rescaling both features to a comparable range [163].

This is already important for RGB images in computer vision applications, where pixel intensity values vary between 0 and 255. Power spectrograms, however, differ fundamentally from images in that their axes (frequency and time) represent different domains, and their value ranges vary significantly due to the logarithmic transformation of power values. Hence, normalisation may play an even larger role when working with power spectrograms for machine learning tasks compared to plain images. Without an appropriate normalisation technique, the network may struggle to learn meaningful patterns from the spectrogram data.

There are also several neural network-based motivations for normalisation of input features:

- Stable gradient flow: Normalised inputs maintain consistent variance across layers, preventing issues such as vanishing or exploding gradients during backpropagation.

This ensures that the network can propagate gradients effectively throughout all layers [164]–[166].

- Reduced saturation: Neural networks often use activation functions like ReLU or tanh, which have flat regions where gradients are near zero. Normalisation keeps input values within the sensitive, non-saturated region of these functions, improving learning [164].
- Faster convergence: By rescaling inputs, normalisation simplifies the optimisation process, allowing gradient descent to converge more efficiently. This can lead to significant reductions in training time [167].

In summary, normalisation ensures balanced feature contributions, stabilises gradient flow, prevents activation saturation, and accelerates convergence. It is especially important for power spectrograms given their unique axes and logarithmic value ranges.

4.2 Overview of normalisation techniques

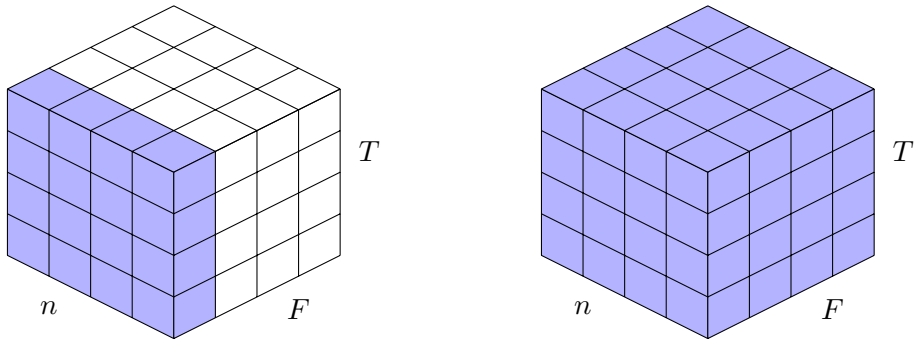
Several normalisation strategies exist, each suited to different data characteristics and modelling requirements. For example, min-max normalisation rescales data to a fixed range, often [0, 1], by mapping the dataset's minimum and maximum values to these bounds. However, this chapter specifically focuses on *standardisation techniques*, a subset of normalisation techniques which rescale data to have a mean of 0 and a standard deviation of 1. These methods are particularly effective for spectrograms with wide dynamic ranges and skewed distributions, as they ensure consistent scaling across the dataset.

Standardisation approaches can be broadly categorised into two groups: global methods, which calculate statistics across the entire dataset, and local methods, which normalise each spectrogram or frequency bin individually. Local normalisation methods, such as those that compute μ and σ for each spectrogram, offer adaptability to unique sample characteristics. However, they risk introducing inconsistencies into the dataset due to wide statistical variations between samples. Moreover, by overfitting to individual spectrogram features, local methods may obscure meaningful global patterns that span the dataset. Given these limitations, we excluded local approaches from our experiments in favour of global techniques, which enforce consistency and preserve cross-sample relationships.

This section examines two widely used global standardisation methods: channel-based normalisation, which independently normalises each frequency bin, and global normalisation, which computes statistics across all time-frequency values in the dataset.

4.2.1 Channel-based normalisation

The channel-based normalisation approach (Figure 4.1a) calculates statistics independently for each frequency bin (or channel) across the entire dataset. For example, for a dataset containing n spectrograms $S_i : i \in [1, n]$, each with dimensions $F \times T$, we would calculate



(a) Channel-based normalisation **(b)** Global normalisation

Figure 4.1: Comparison of global and channel-based normalisation techniques. The blue cubes represent the subsets of the spectrogram used for averaging. Figure inspired by [165, Fig. 2].

the average and standard deviation of each frequency bin $f \in F$ across all spectrograms; that is:

$$\mu_f = \frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} S_i(f,t) \tag{4.1}$$

$$\sigma_f = \sqrt{\frac{1}{nT} \sum_{i=1}^{n} \sum_{t=1}^{T} (S_i(f,t) - \mu_f)^2} \tag{4.2}$$

Each value within a frequency bin is then normalised using these statistics:

$$S(f,t) = \frac{S(f,t) - \mu_f}{\sigma_f}. \tag{4.3}$$

This method treats each frequency bin as an independent channel, analogous to how RGB channels are normalised in image data. Channel-based normalisation is particularly effective when frequency bins exhibit distinct statistical distributions. For example, Ruffini et al. employed channel-normalised spectrograms derived from 14-channel electroencephalography (EEG) data to address such channel-specific variability [168].

4.2.2 Global normalisation

The global normalisation method (Figure 4.1b) computes the mean and standard deviation across all time-frequency values across all spectrograms of the dataset. That is, for a dataset containing n spectrograms $S_i : i \in [1, n]$, each with dimensions $F \times T$:

$$\mu_{\text{global}} = \frac{1}{nFT} \sum_{i=1}^{n} \sum_{f=1}^{F} \sum_{t=1}^{T} S_i(f,t) \tag{4.4}$$

$$\sigma_{\text{global}} = \sqrt{\frac{1}{nFT} \sum_{i=1}^{n} \sum_{f=1}^{F} \sum_{t=1}^{T} (S_i(f,t) - \mu_{\text{global}})^2} \tag{4.5}$$

Each spectrogram S is then normalised using these global statistics:

$$S(f,t) = \frac{S(f,t) - \mu_{\text{global}}}{\sigma_{\text{global}}} \tag{4.6}$$

By rescaling all spectrogram values to have a mean of 0 and a standard deviation of 1, global normalisation reduces variability and standardises the data. This can improve a model's ability to learn from the inputs, as highlighted by Kroenke in their review of normalisation techniques [169]. For example, Primus and Widmer normalise their log-Mel spectrograms using global normalisation prior to feeding them into an audio spectrogram transformer model [170].

The comparison of original, global-normalised, and channel-normalised spectrograms, as shown in Figure 4.2, reveals minimal visual differences in the structural patterns across the three techniques. This outcome is expected, as the primary purpose of normalisation is not to alter the fundamental structure of the spectrogram but to rescale their amplitudes into a standardised range. The key difference in Figure 4.2a lies in the colour bar scaling: in the original spectrogram, amplitude values span a range of approximately -30 to 30 dB, whereas in the global and channel-normalised spectrograms, the values are rescaled to a narrower range, approximately -3 to 4 dB. This is further supported by Figure 4.2c which highlights the rescaling of amplitudes for a single random time segment in each spectrogram. Additionally, the amplitude histograms in Figure 4.2b show that while the overall distribution shapes remain similar, the normalised spectrograms exhibit distributions centred around zero. This comparison highlights a key property of normalisation: the normalised spectrograms still retain the important information necessary for classification while standardising the data for greater gradient stability during machine learning tasks.

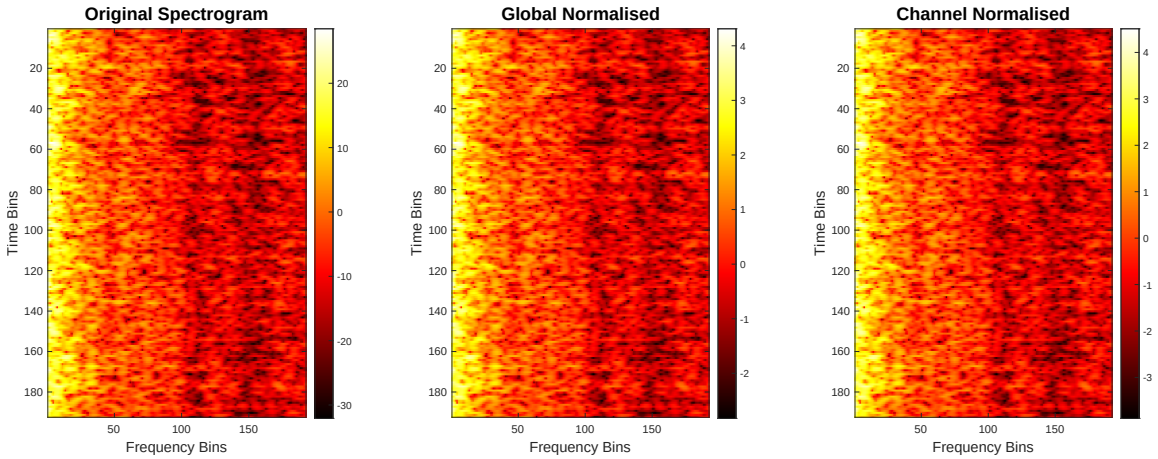
4.3 Experiments

To evaluate the impact of global and channel-based normalisation on the DeepShip dataset, we designed an experiment to compare the performance of our benchmark CNN-LSTM model across three scenarios: baseline (no normalisation), global normalisation, and channel-based normalisation. The objective was to isolate the influence of these normalisation methods on model accuracy.

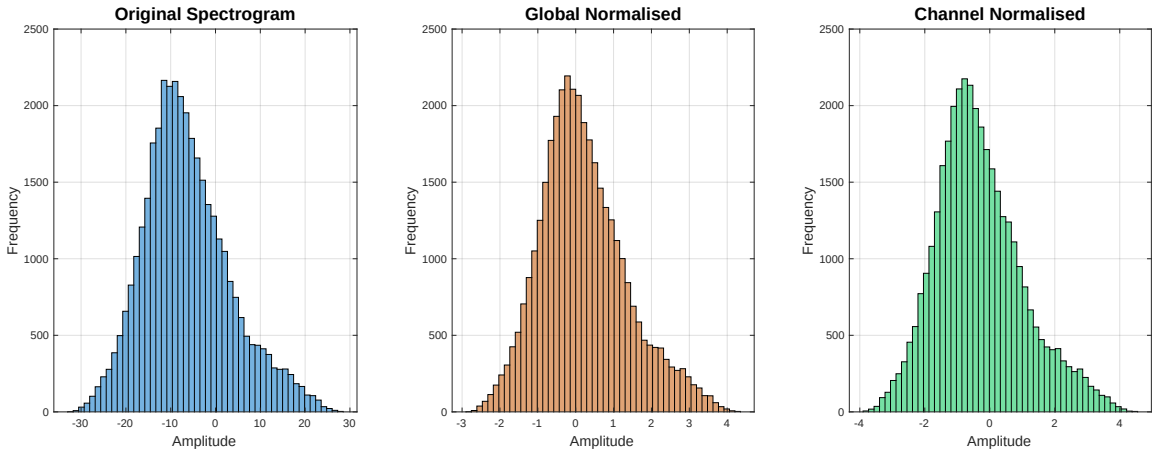
4.3.1 Methodology

Normalisation was implemented as a two-pass operation in MATLAB. In the first pass, the required statistics for normalisation were computed:

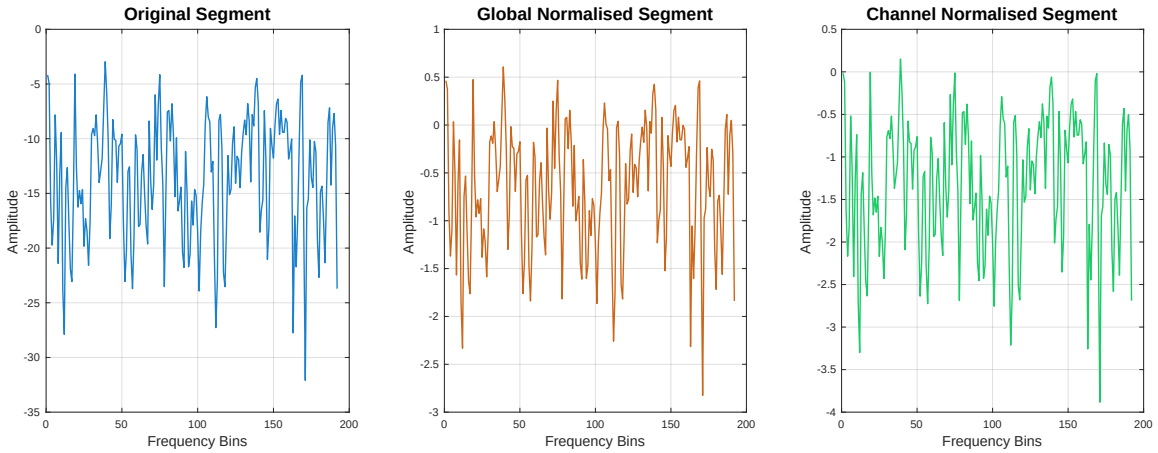
- For global normalisation, the mean and standard deviation were calculated across all time-frequency bins in the dataset.
- For channel-based normalisation, these statistics were calculated separately for each frequency channel.



(a) Comparison of original spectrogram with global-normalised and channel-normalised spectrograms.



(b) Comparison of amplitude histograms for original, global-normalised, and channel-normalised spectrograms.



(c) Comparison of a random time segment for original, global-normalised, and channel-normalised spectrograms.

Figure 4.2: Visual comparison of normalisation methods.

In the second pass, these precomputed statistics were used to apply the respective normalisation transformations to each spectrogram. The spectrograms were then exported as `.mat` files to seperate directories.

The benchmark CNN-LSTM model was trained on each type of spectrogram under identical conditions. The same 10-fold cross-validation split was used across all experiments to ensure a consistent and fair comparison. The same hyperparameters and training configuration was used for all experiments, as outlined in Table 3.3 and Section 3.3.2. Model performance was evaluated using accuracy as the primary metric, and training and validation loss curves were recorded to analyse convergence trends qualitatively.

To ensure reproducibility and consistency, all experiments were conducted using MATLAB version 2024a and Keras 2.10, with a seeded random number generator and GPU-accelerated training.

4.3.2 Results

The results of this experiment, summarised in Table 4.1, show minimal difference between the three normalisation strategies. The baseline model previously achieved an accuracy of 63.41%, while both global and channel-based normalisation resulted in lower accuracies of 62.99% and 63.16% respectively. Training and validation accuracy-loss curves are shown in Figure 4.3 and 4.4.

Table 4.1: Comparison of normalisation strategies on the DeepShip dataset.

Normalisation strategy	Accuracy (%)	Precision
Global normalisation	62.99	65.66
Channel-based normalisation	63.16	66.34
Baseline	**63.41**	**66.53**

4.3.3 Discussion

The lack of improvement in classification accuracy was unexpected, particularly given the extensive body of research demonstrating the theoretical and practical benefits of normalisation for machine learning tasks ([163], [165], [169]–[171]).

The most plausible explanation for this null result lies in the relatively consistent scale and distribution of spectrogram features in the DeepShip dataset, influenced by two key factors: (a) the logarithmic transformation applied during the conversion to power spectrograms, and (b) the consequences of the dataset's recording setup.

The conversion of amplitude spectrograms into power spectrograms involved a base-10 logarithmic transformation (Section 3.2). This process inherently standardises the input values by compressing their dynamic range, reducing the variability in feature scales, and possibly diminishing the added benefits of explicit normalisation.

A more fundamental factor, however, may be the controlled recording setup used to create the DeepShip dataset. As described in Section 2.3.4, the DeepShip recordings were collected using a fixed hydrophone located approximately 145 metres below sea level (Figure 2.9). Recordings were only made when a single vessel was within a 2km radius of the hydrophone, ensuring a relatively uniform acoustic environment. This controlled setup likely causes the DeepShip dataset to have an internally consistent structure across recordings. Consequently, normalisation techniques that target variability in feature scales may not be as impactful for the DeepShip dataset, as compared to datasets collected using more dynamic setups, such as towed arrays travelling through a variety of acoustic environments.

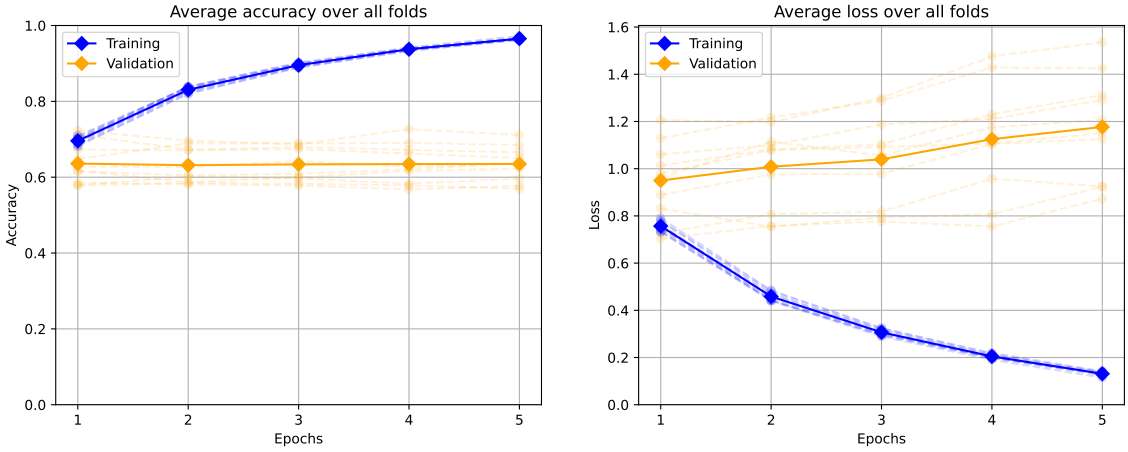
Additionally, it is possible that the relatively short training duration of five epochs limited the ability of the model to fully leverage any benefits introduced by normalisation. Normalisation's primary advantages, such as stabilising gradient flow and accelerating convergence, may become more evident over longer training periods or with deeper models.

A key limitation of the experiment was the approach taken to calculate the normalisation statistics. In standard practice, the mean and standard deviation are calculated on the training set and then applied to both the training and test sets. This ensures that the test data remains unseen during the calculation of normalisation parameters, preserving the integrity of the evaluation process and preventing data leakage. However, due to the k-fold cross-validation setup used in this study, where training and testing sets change dynamically in each fold, it was impractical to calculate the mean and standard deviation exclusively from the training set. Instead, normalisation statistics were computed using the entire dataset. While this compromise was necessary given the experimental constraints, it may have compromised the validity of the experiment by introducing a minor data leakage effect, which could distort the results and undermine the reliability of the evaluation process.

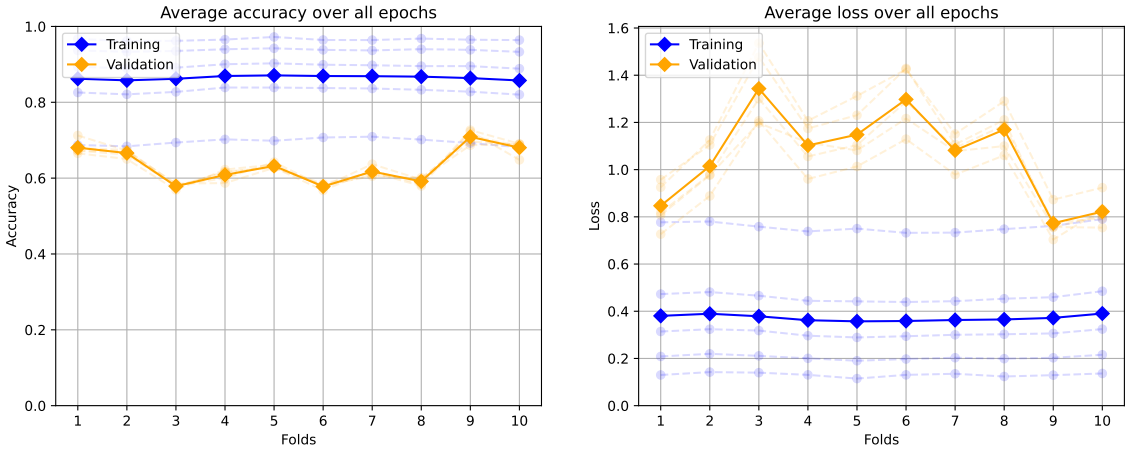
The training and validation accuracy-loss curves for both the channel normalisation experiment (Figure 4.3) and global normalisation experiment (Figure 4.4) show a great similarity between not only each other but also the benchmark model's accuracy-loss curves (Figure 3.9), signifying that the training procedure executed as expected with no major errors. In fact, the validation loss curve for the channel normalisation experiment (Figure 4.3a) demonstrates a lower variance across folds compared to the baseline model, potentially indicating a slight improvement in the model's generalisability.

4.3.4 Conclusion

This experiment demonstrated that global and channel-based normalisation had minimal impact on classification performance for the DeepShip dataset, with results obtained comparable to the baseline (unnormalised) model. The minimal improvement suggests that the preprocessing steps already applied – such as the power spectrogram logarithmic conversion – already played a significant role in normalising the input data, diminishing the additional benefits explicit standardisation was expected to offer. Additionally, the controlled recording setup of the DeepShip dataset, including the use of a fixed hydrophone

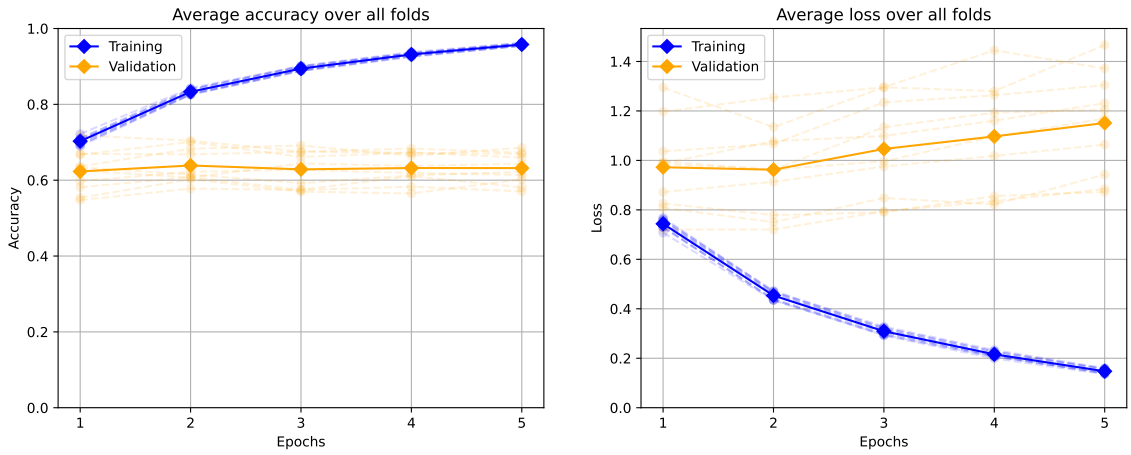


(a) Training and validation accuracy and loss trends per epoch for the channel normalisation experiment.

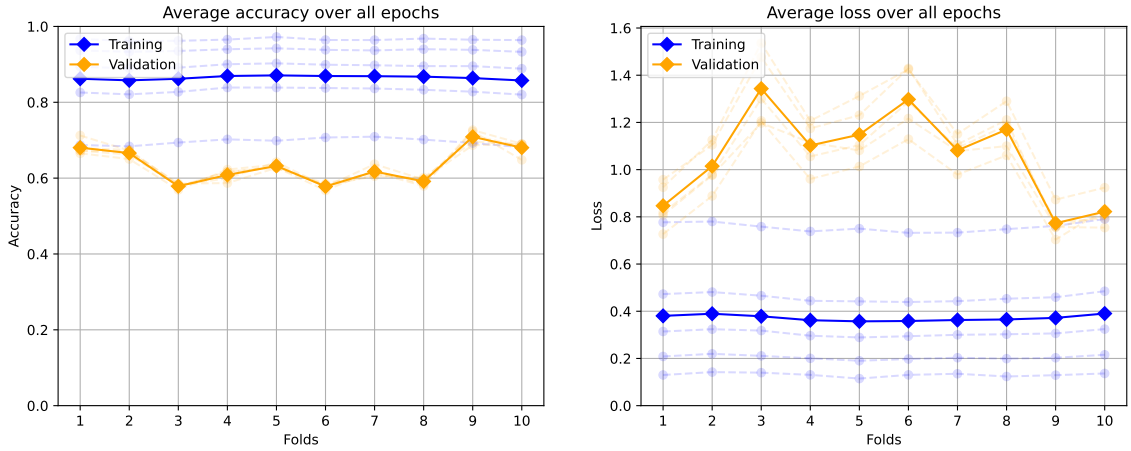


(b) Training and validation accuracy and loss trends per fold for the channel normalisation experiment.

Figure 4.3: Comparison of training and validation performance for channel normalisation, showing accuracy and loss curves evaluated by (a) epoch and (b) fold.



(a) Training and validation accuracy and loss trends per epoch for the global normalisation experiment.



(b) Training and validation accuracy and loss trends per fold for the global normalisation experiment.

Figure 4.4: Comparison of training and validation performance for global normalisation, showing accuracy and loss curves evaluated by (a) epoch and (b) fold.

and uniform recording conditions, likely contributed to the consistent feature scales observed. While normalisation may still hold value for underwater acoustics, its impact is likely to be more pronounced in datasets with greater variability, such as those collected using dynamic towed arrays in diverse acoustic environments.

These findings highlight the importance of aligning preprocessing strategies with the specific characteristics of a dataset. Future work could explore alternative normalisation techniques, such as 0-1 normalisation or local normalisation, and examine their effects over longer training cycles or their role in improving training stability and convergence. Researchers may wish to begin with amplitude spectrograms rather than power spectrograms to isolate the impact of normalisation without other preprocessing factors influencing the results, and may wish to explore the effect of normalisation using alternative datasets which exhibit greater variability in their recording setup. Finally, while normalisation did not significantly impact classification accuracy in this study, it may still offer other advantages not captured in this experiment, such as enabling more efficient training. Future work could explore these aspects more thoroughly in order to gain a better understanding of the role of normalisation in the field of underwater acoustic target recognition.

Chapter 5

Detrending

5.1 Introduction

Detrending is a signal processing technique aimed at removing long-term trends from data to isolate fluctuations that are more relevant for analysis. In the context of underwater acoustic target recognition, detrending can suppress broadband noise and highlight transient features, such as the periodic signals generated by ship machinery, which are critical for classification tasks. By removing these underlying trends, the remaining signal may provide cleaner, more interpretable inputs for machine learning models.

This chapter explores the application of ℓ_1 detrending, a robust technique that builds on traditional methods like the Hodrick-Prescott filter. The goal of this experiment is to evaluate whether ℓ_1 detrending can enhance the performance of the baseline CNN-LSTM model by improving the clarity of spectrogram features used for classification. Through this investigation, we seek to address a fundamental question: does detrending improve machine learning model performance in the field of underwater acoustics, or does it inadvertently remove essential features needed for accurate classification?

5.2 Overview of detrending algorithms

Detrending algorithms aim to remove underlying trends from data, isolating the fluctuations that are more relevant for analysis. Mathematically, we can describe this as idea as an optimisation problem. We are given a scalar time series y_t, $t = 1, \ldots, n$, assumed to consist of an underlying slowly varying trend x_t, and a more rapidly varying random component z_t. Our goal is to estimate the trend component x_t, or equivalently, estimate the random component $z_t = y_t - x_t$.

The simplest detrending method involves differencing the data. For a dataset with points $(x_1, x_2, \ldots, x_n)$, detrending by differencing computes a new dataset as $(x_2 - x_1, x_3 - x_2, \ldots, x_n - x_{n-1})$. While straightforward, this method can be overly sensitive to noise, as it does not distinguish between the trend and high-frequency fluctuations.

Another widely used approach is to detrend by fitting a model to the data and removing the fitted trend. A simple example is linear detrending, where a line is fit to the data

via least squares regression, and the residuals make up the detrended result. More complex versions of this technique involve fitting higher-order polynomials or other parametric models to capture nonlinear trends.

One of these more sophisticated methods is the Hodrick-Prescott (H-P) filter, a technique popularised in the 1990s originally for use in economics [172], [173]. The H-P filter is widely used in time-series analysis to decompose a signal into its trend and cyclical components. It does so by solving an optimisation problem that minimises a loss function, balancing two competing objectives: fitting the trend closely to the data and ensuring the smoothness of the trend. Specifically, the H-P filter minimises:

$$\sum_{t=1}^{n}(y_t - x_t)^2 + \lambda \sum_{t=2}^{n-1}(x_{t-1} - 2x_t + x_{t+1})^2 \tag{5.1}$$

where y_t is the observed signal, x_t is the trend, and λ is the smoothing parameter. The first term enforces the trend's closeness to the original data, while the second term penalises sharp changes in the trend to ensure smoothness.

The smoothing parameter λ plays a crucial role: larger λ values result in smoother trends, while smaller λ values retain more of the original signal's variability. While the H-P filter is effective for detecting long-term trends in economic data, it has limitations. It assumes that the trend is smooth, making it less effective for signals with sharp changes or localised features. Additionally, its reliance on the ℓ_2 norm in the smoothness term can make it sensitive to outliers which can disproportionately affect the squared differences.

5.2.1 ℓ_1 Detrending Algorithm

The ℓ_1 detrending algorithm [174] builds on the principles of the H-P filter but replaces the ℓ_2 norm in the smoothness penalty with the ℓ_1 norm. This modification makes the ℓ_1 algorithm more robust to outliers and better suited to capturing sharp changes in the trend. The optimisation problem for ℓ_1 detrending is expressed as:

$$\frac{1}{2}\sum_{t=1}^{n}(y_t - x_t)^2 + \lambda \sum_{t=2}^{n-1}\|x_{t-1} - 2x_t + x_{t+1}\| \tag{5.2}$$

Here, the ℓ_1 norm ($\|\cdot\|$) in the second term penalises large differences between successive points in the trend, enforcing smoothness while preserving sharp transitions.

The ℓ_1 detrending algorithm retains the flexibility of the H-P filter with its tunable regularisation parameter λ, but its use of the ℓ_1 norm allows it to better handle datasets with localised events or abrupt changes. Ship radiated noise, for instance, often contains periodic narrowband features from machinery noise, which can be masked by the broadband components in the signal. The ℓ_1 algorithm's sensitivity to these features makes it particularly suitable for UATR tasks. By fine-tuning λ, we can strike a balance between removing the trend and preserving the narrowband features critical classification.

5.3 Experiments

To evaluate the impact of detrending on underwater acoustic target recognition, we designed an experiment to assess the performance of our benchmark CNN-LSTM model with spectrograms processed under varying levels of ℓ_1 detrending. By removing long-term trends and broadband noise from the spectrograms, the objective of this experiment is to determine whether detrending enhances classification accuracy by isolating meaningful features while suppressing irrelevant signal components.

5.3.1 Methodology

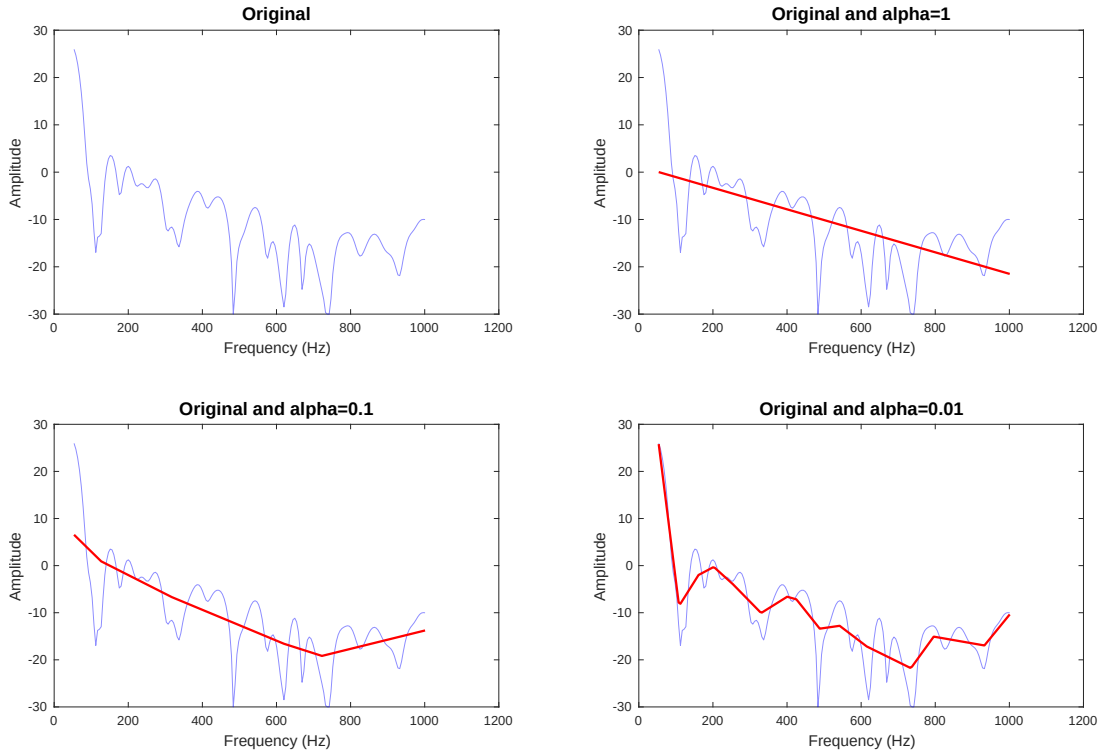
The ℓ_1 detrending algorithm was implemented in MATLAB as a modular and flexible function which builds upon the original code published by the authors. The primary objective of this implementation was to enable experimentation with different detrending strengths through tweaking the regularisation parameter λ, to which we assign a coefficient α, as well as provide visualisations to evaluate the effectiveness of detrending for classification and image segmentation tasks.

Recall that the detrending algorithm is formulated as an optimisation problem where we try to balance two objectives: minimising the residual between the input signal and its estimated trend while ensuring that the trend remains smooth. The regularisation parameter λ governs this trade-off. For any given signal, $\lambda_{\max}$ is the value of λ beyond which the algorithm will simply return a 1-to-1 translation, or *affine fit*, for the data. This upper bound is computed using a function provided by the authors, `l1tf_lambdamax`. The value of λ used during detrending is then defined as a fraction of $\lambda_{\max}$, controlled by our assigned coefficient α. By experimenting with various values of α, it is possible to fine-tune the balance between trend smoothness and residual suppression.

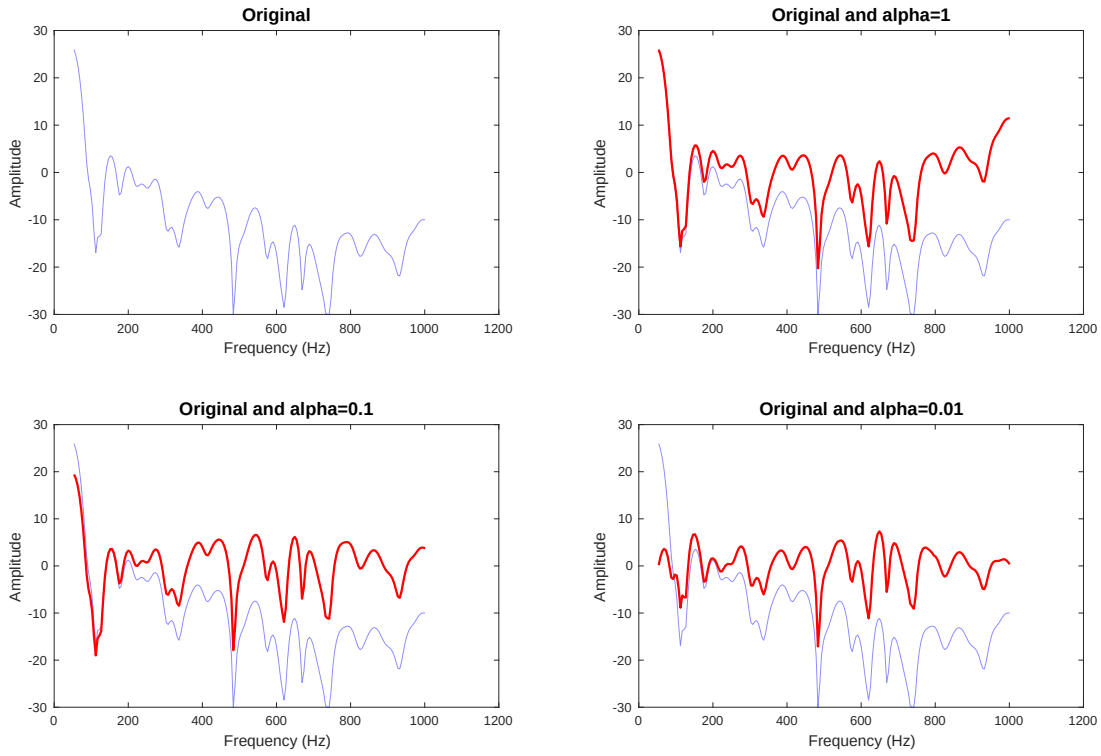
To support this experimentation, our implementation of the ℓ_1 detrending algorithm allows the user to specify up to 10 values of α in a single run. The function computes detrended spectrograms for each specified α value and generates a range of diagnostic plots that facilitate analysis.

1. Segment trend plot: This plot (Figure 5.1a) overlays a single random time segment with its corresponding ℓ_1 trends for various α values.

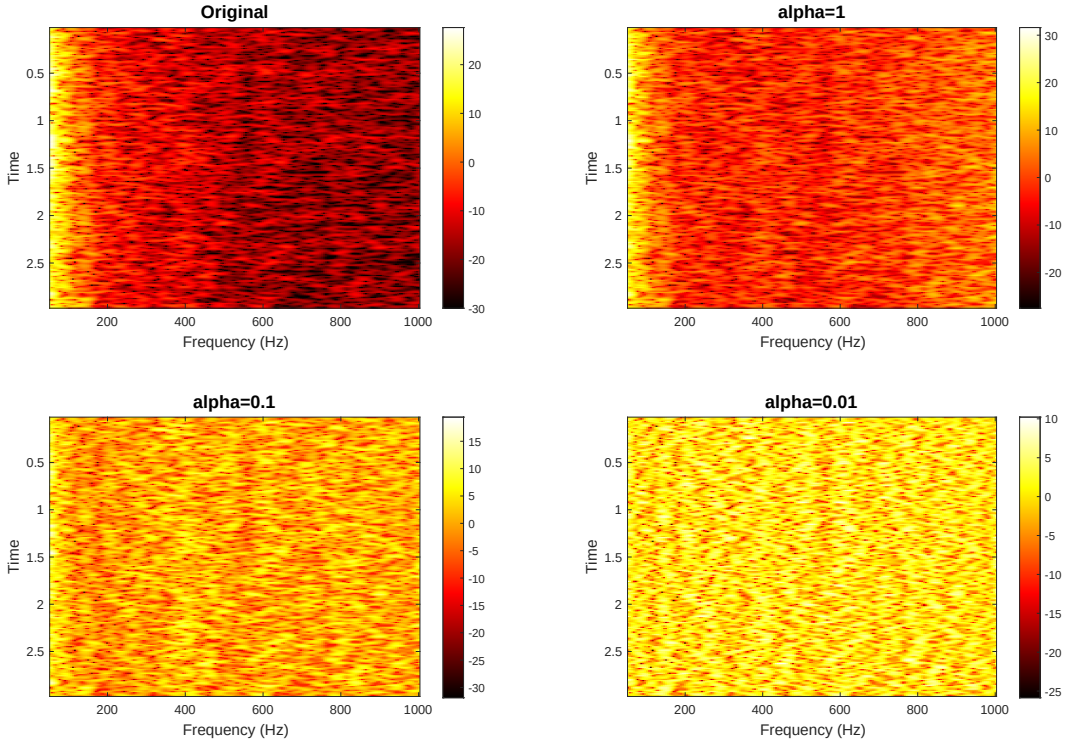
2. Segment detrended plot: This diagnostic (Figure 5.1b) shows the same time segment before and after detrending for various α values. It is the most helpful diagnostic plot to visualise the changes introduced by detrending, especially the removal of long-term trends such as those created by background noise. By examining the degree of smoothing introduced for different α, we could evaluate how effectively the algorithm removed noise without discarding important signal features. Smaller α values were observed to retain more signal detail but were less effective at suppressing noise, while larger α values removed noise more aggressively but often smoothed out transient features.



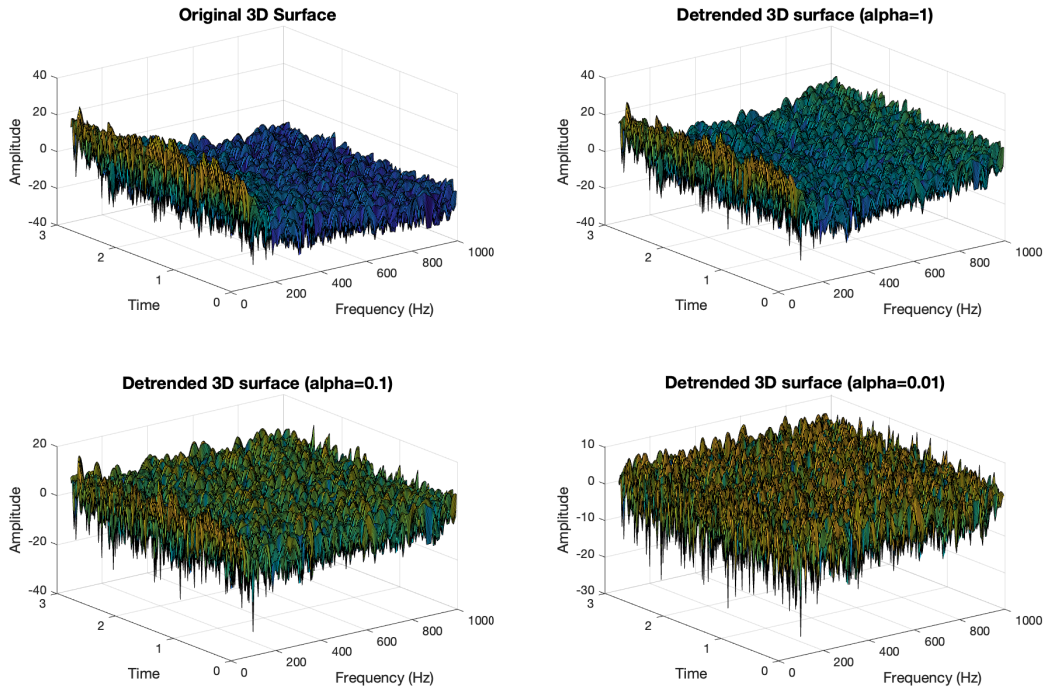
(a) Overlay of a random time segment with its corresponding ℓ_1 trend at various α values.



(b) Random time segment before and after detrending at different α values, highlighting the removal of long-term trends in the signal.



(c) Comparison of original and detrended spectrograms produced using ℓ_1 detrending at various α values.



(d) Three-dimensional surface plots of the original spectrogram and detrended spectrograms at different α values.

Figure 5.1: Visual analysis of ℓ_1 detrending for various α values, illustrating its impact on spectrograms through comparative visualisations, time-segment overlays, and 3D surface plots.

3. Spectrogram comparison: This diagnostic plot (Figure 5.1c) presents the original spectrogram alongside detrended spectrograms generated using different α values. It provides an overview of how detrending affects the overall amplitude structure, particularly in regions dominated by long-term trends or broadband noise. By visually assessing these plots, we could identify the boundaries where detrending was either too aggressive (overfitting) or too weak (underfitting).

4. 3D surface plots: These 3D plots (Figure 5.1d) provide an alternative representation of how detrending impacts amplitude values across time and frequency.

Using these diagnostic plots, we determined that α values in the range $[10^{-3}, 1]$ provided a suitable balance between noise suppression and signal preservation. Hence, we chose three such α values for further experimentation: 10^{-2}, 10^{-1}, and 1. These values were chosen in hopes of capturing a wide spectrum of detrending strengths, ranging from subtle noise suppression to more aggressive trend removal.

Then, to evaluate the effect of ℓ_1 detrending on UATR classification performance, the baseline 3-second DeepShip spectrograms (Section 3.2) were processed using the three selected α values (10^{-2}, 10^{-1}, and 1). Each detrended spectrogram was saved to a separate directory as a `.mat` file and subsequently used as input to the CNN-LSTM baseline model.

The CNN-LSTM model was trained under identical conditions to the baseline model to ensure fair comparisons. The same architecture, hyperparameters, and training configuration were used for all experiments, as detailed in Table 3.3 and Section 3.3.2, and the same 10-fold cross-validation splits used in previous experiments were applied to ensure consistency and comparability across results. Model performance was evaluated using accuracy as the primary metric, and training and validation loss curves were recorded to analyse convergence trends qualitatively.

To further ensure robustness, all experiments were conducted using MATLAB version 2024a and Keras 2.10, with a seeded random number generator and GPU-accelerated training. This setup, consistent with the baseline model, guarantees that any observed differences in model performance can be attributed solely to the effect of ℓ_1 detrending and its parameter variations.

5.3.2 Results

The classification results for the ℓ_1 detrending experiments, presented in Table 5.1, show that all three detrending configurations resulted in lower classification accuracy and precision compared to the baseline. The baseline model achieved an accuracy of 63.41% and a precision of 66.53%, whereas the detrended models showed a consistent decline, with accuracy reductions ranging from 15.19% at $\alpha = 10^{-2}$ to 7.78% at $\alpha = 0.1$. Precision followed a similar trend, with decreases across all α values. Training and validation accuracy-loss curves are shown in Figure 5.2 and 5.3.

Table 5.1: Classification results using ℓ_1 detrending algorithm at various α.

Detrending parameter	Accuracy (%)	Precision (%)
$\alpha = 10^{-2}$	48.22	59.09
$\alpha = 10^{-1}$	52.03	62.54
$\alpha = 1$	55.63	63.97
Baseline (no detrending)	**63.41**	**66.53**

5.3.3 Discussion

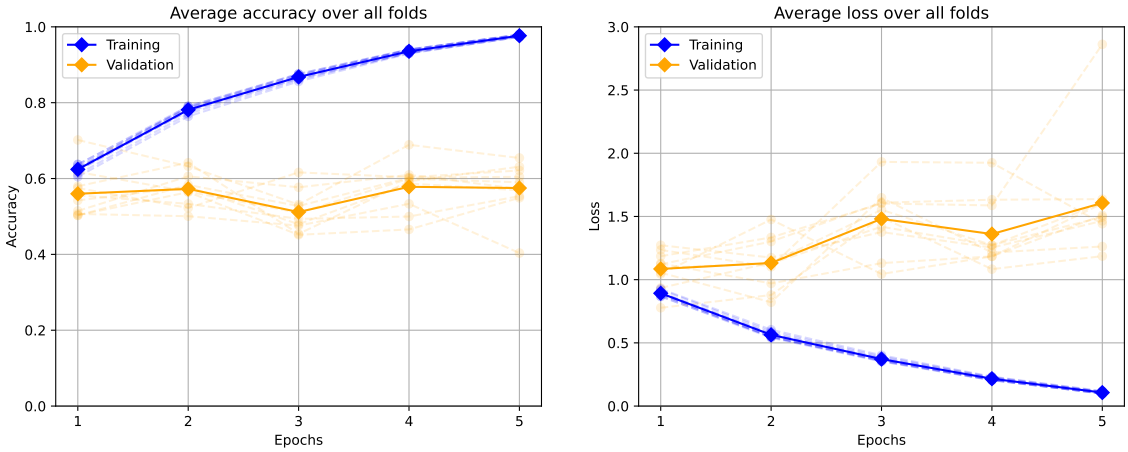
The consistent drop in accuracy across all α values suggests that ℓ_1 detrending, while effective at removing long-term trends, may also be inadvertently removing or distorting features essential for classification.

Lower α values may be causing *over-smoothing*; that is, the ℓ_1 algorithm could be aggressively suppressing broadband noise through the removal of the long-term trend, but also smoothing out transient narrowband features such as machinery noise which are key discriminators for different vessel types. On the other hand, higher α values may be preserving finer details while failing to adequately suppress broadband noise: a phenomenon called *under-suppression*.

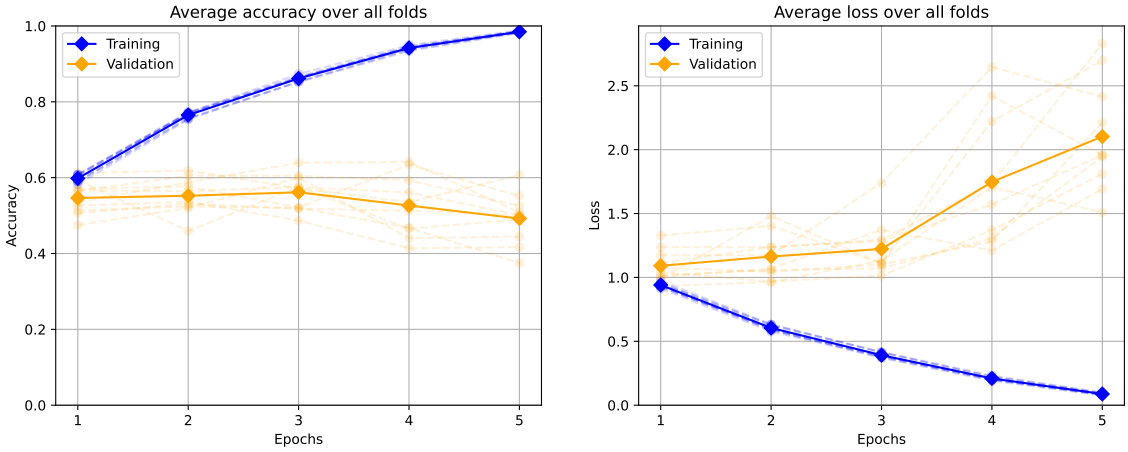
It is also possible that the short 3 second duration of the input segments (Section 3.1.2) is preventing the ℓ_1 algorithm from accurately capturing the broader trend of each recording. This could lead to incomplete or imprecise detrending.

The poor results may further stem from the combined effect of using a detrending algorithm, which alters the spatial and temporal characteristics of the data, in tandem with a CNN-LSTM classifier, which heavily relies on these characteristics to perform classification (in fact, this was the primary reason for choosing this model; see Section 3.3). The ℓ_1 algorithm may have disrupted the input, which is reflected in the instability of the accuracy-loss curves for the detrending experiments (Figures 5.2 and 5.3). In contrast to the uniform curves obtained for the baseline model (described in Section 3.4), the accuracy-loss curves for the detrending experiments reveal significant irregularities. In Figure 5.2, the validation curves for individual folds deviate notably from the average trend, with erratic fluctuations that suggest inconsistent convergence. This issue is even more pronounced in Figure 5.3b, where the validation loss curves for each epoch show dramatic variability. Such behaviour is indicative of challenges in the model's ability to learn and generalise effectively, suggesting that the DeepShop spectrograms may have lost important information in the detrending process which the CNN-LSTM model relied on for accurate classification. As a result, the model likely struggled to interpret the input features in a coherent way, leading to unstable performance and poor generalisation across folds.

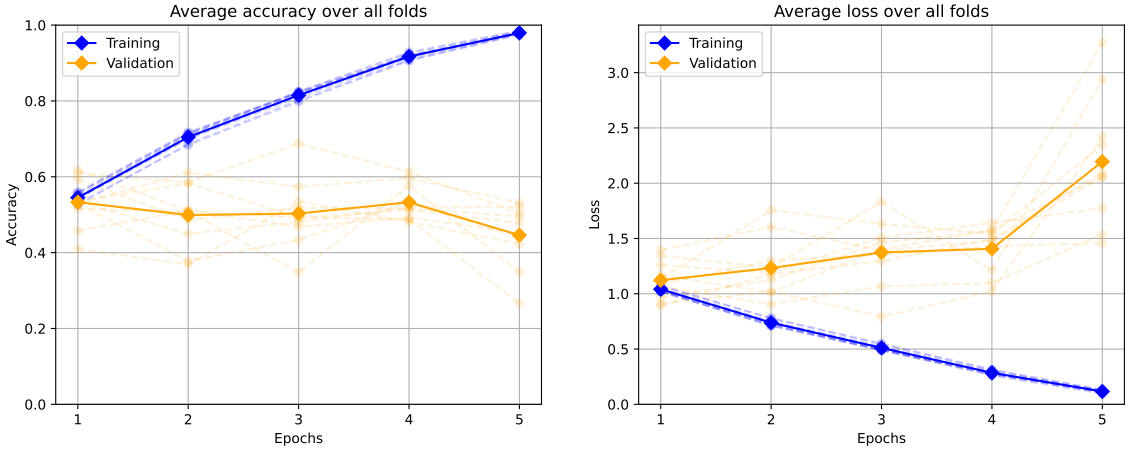
Fundamentally, these findings raise questions about the appropriateness of detrending for UATR tasks. While detrending is commonly used in domains like finance or EEG analysis, where trends often obscure the signal of interest, it may simply be that through



(a) Validation accuracy and loss by epoch for ℓ_1 detrending with $\alpha = 1$.

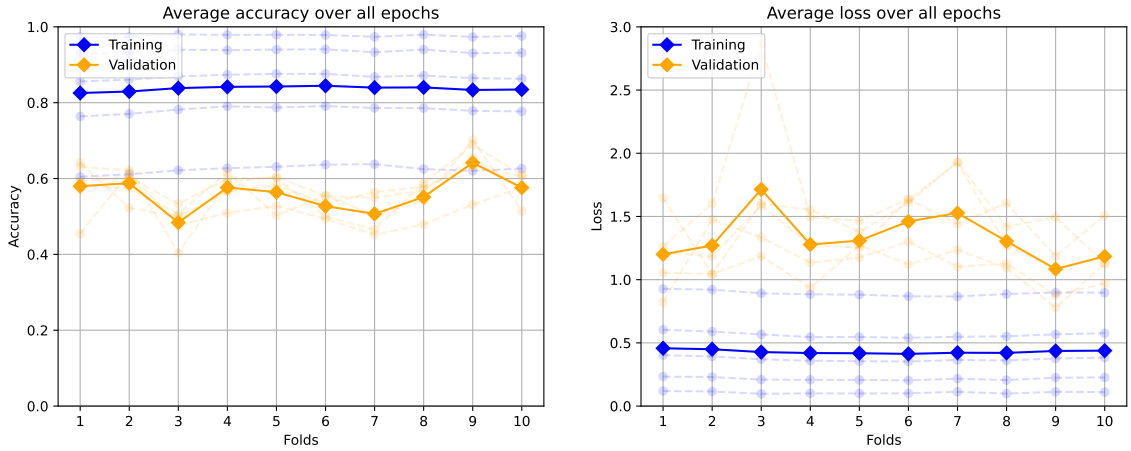


(b) Validation accuracy and loss by epoch for ℓ_1 detrending with $\alpha = 10^{-1}$.

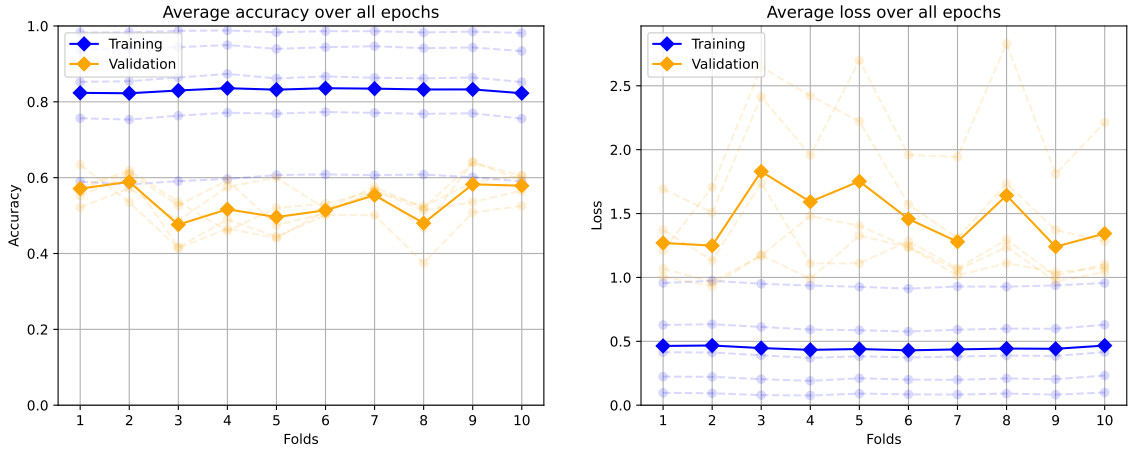


(c) Validation accuracy and loss by epoch for ℓ_1 detrending with $\alpha = 10^{-2}$.

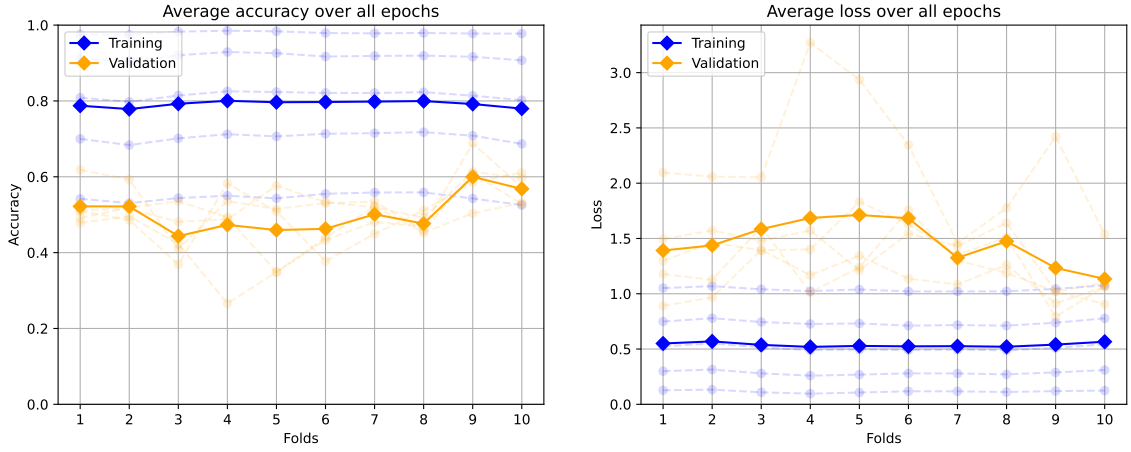
Figure 5.2: Validation accuracy and loss curves for ℓ_1 detrending experiments by epoch. Faint lines show curves for individual folds while the dark line shows the average across all folds.



(a) Validation accuracy and loss by fold for ℓ_1 detrending with $\alpha = 1$.



(b) Validation accuracy and loss by fold for ℓ_1 detrending with $\alpha = 10^{-1}$.



(c) Validation accuracy and loss by fold for ℓ_1 detrending with $\alpha = 10^{-2}$.

Figure 5.3: Validation accuracy and loss curves for ℓ_1 detrending experiments by fold. Faint lines show curves for individual epochs while the dark line shows the average across all epochs.

removing long-term trends, the detrended spectrograms are losing the distinct structure necessary for the baseline CNN-LSTM model to effectively discern between vessel types.

5.3.4 Conclusion

While ℓ_1 detrending holds theoretical promise for its ability to suppress long-term trends, its practical application to UATR tasks presents significant challenges. The classification performance of our baseline CNN-LSTM model declined when using ℓ_1-detrended spectrograms across all tested α values, with accuracy and precision reductions suggesting the unintended removal or distortion of essential spectrogram features. Lower α values appeared to over-smooth transient features critical for vessel classification, while higher α values failed to adequately suppress broadband noise. Furthermore, the erratic accuracy-loss curves observed during training suggest that the detrending process disrupted the spatial and temporal structure of the spectrograms, hindering the CNN-LSTM model's ability to converge effectively.

Future work should investigate these effects more thoroughly. First, further investigation into the interaction between detrending and the CNN-LSTM architecture is needed to better understand the underlying causes of the poorly-behaved accuracy and loss training curves. It may also be valuable to explore the effects of detrending on other model architectures to assess whether the observed issues are specific to the CNN-LSTM framework. Additionally, experimenting with a wider range of α values could help identify configurations that strike a better balance between noise suppression and the preservation of key features. Extending the length of the input audio segments could also allow the detrending algorithm to capture long-term trends more accurately. Finally, a comparison of ℓ_1 detrending to other approaches, such as wavelet-based detrending, could provide insights into whether improved performance is achievable for this dataset. Ultimately, while ℓ_1 detrending offers potential in other domains, its current application to the field of underwater acoustic target recognition requires further refinement.

Chapter 6

Denoising

Denoising refers to the process of removing unwanted noise from a signal while still retaining its most meaningful components. In the context of underwater acoustic spectrograms, this involves isolating ship radiated noise or narrowband events from background noise such as wind, waves, or marine life. In this chapter we aim to experiment with applying various image denoising methods to underwater acoustic data.

6.1 Introduction

The primary objective of denoising is to recover an underlying clean signal y from its observed noisy recording x. Mathematically, the observed signal can be expressed as:

$$x = y + n \tag{6.1}$$

where n represents the noise contaminating the signal. Recovering y from x is a challenging problem due to the stochastic nature of n and the lack of prior knowledge about the statistical distribution of either n or y. Traditional denoising methods rely on explicit models of noise, such as Gaussian or Poisson distributions, to model the relationship between the observed noisy signal x and the true clean signal y. This allows them to approximate the likelihood $p(x \mid y)$, which represents the probability of observing the noisy signal x given that the true signal is y. By doing this, they derive an optimal denoising function $f(x)$ that estimates y from x:

$$f(x) = \hat{y} \approx y \tag{6.2}$$

where $\hat{y}$ is the estimated clean signal.

This denoising process is particularly important when dealing with underwater acoustic signals as underwater environments are inherently noisy, with ambient disturbances arising from both natural sources, such as wind, waves, rainfall, and marine life, as well as anthropogenic sources such as ship engines, industrial activities, and even self-noise from acoustic detection equipment (Section 2.1.3). Such noise sources mask or distort the meaningful components of underwater acoustic signals, significantly lowering their signal-to-noise ratio and hindering subsequent analysis tasks such as detection, recognition, localisation,

and tracking of underwater targets [175]–[177]. By removing background noise, denoising enhances the visibility of target signals, enabling more accurate feature extraction and analysis [178]–[180]. For example, in underwater biology research, denoising is first applied to minimise environmental interference when analysing marine animal echolocation signals, allowing for more accurate signal analysis [181].

The importance of denoising is further amplified by advances in stealth technology and the use of noise jamming techniques by modern vessels. These strategies are designed to obscure acoustic signatures, making it increasingly difficult for third parties to capture unambiguous ship radiated noise signals [182]. Furthermore, the presence of inherent self-noise in hydrophones and other underwater detection equipment only further complicates this challenge. As a result, the development of robust noise reduction methods has become essential to mitigate these diverse noise sources and improve the reliability and accuracy of underwater acoustic systems.

6.1.1 Challenges in denoising underwater acoustic signals

Denoising underwater acoustic signals is a complex task due to the unique challenges posed by the underwater environment and the inherent limitations of available data and techniques. This section briefly delves into some of the most prominent challenges faced in denoising underwater acoustic signals.

Lack of hydrophone array recordings Many underwater acoustic target recognition datasets such as DeepShip (Section 2.3.4) and ShipsEar (Section 2.3.4) consist of recordings from single hydrophones rather than arrays. This restricts the use of spatial denoising techniques such as beamforming, which rely on multiple hydrophones to isolate and suppress noise from specific directions. As a result, research is often limited to spectral or temporal approaches, which are not as robust, accurate, or reliable as spatial denoising techniques using hydrophone arrays.

Complex noise patterns and interactions Noise in underwater environments is highly variable and includes a wide range of frequencies, amplitudes, and temporal characteristics. For example, biological noise from marine life (snapping shrimp, dolphins, etc.) is transient and broadband, while geophysical noise (e.g. wind, waves, and precipitation) contributes to low-frequency background noise [183]. These diverse noise sources interact in non-linear, non-Gaussian, and non-stationary ways, making it difficult to develop universal noise models or apply traditional denoising techniques effectively [180], [182].

Variability of the acoustic environment The underwater acoustic environment is inherently dynamic, further complicating denoising tasks. Noise levels can vary dramatically throughout the day, with phenomena such as the morning chorus of snapping shrimp only occurring at specific times, or shifting wave conditions due to tides altering noise characteristics. Environmental factors, including changes in wind speed, water temperature,

salinity, and depth, affect sound propagation and create variable conditions that denoising algorithms must adapt to. Noise levels and types also differ significantly across locations due to local marine life activity, human presence, and geophysical factors. This variability makes it challenging to design algorithms that can generalise across diverse acoustic environments.

Overlapping characteristics of noise and target signal A fundamental challenge in underwater acoustic denoising lies in defining what constitutes "noise." Misclassification can lead to the unintentional removal of valuable signal components, compromising the integrity of the denoising process. For example, while certain types of noise, such as coloured noise (e.g., white, pink, brown noise), have well-defined spectral characteristics, some target signals can share similar frequency patterns. This overlap between the noise and the signal complicates the task of distinguishing them accurately, making it harder to develop effective denoising techniques.

6.1.2 Conventional techniques and the potential of deep learning

Traditional digital signal processing techniques, such as wavelet transforms, empirical mode decomposition, and variational mode decomposition, are well-established methods in underwater acoustic signal denoising. These methods rely on the principles of statistical signal processing, leveraging assumptions about the spectral or temporal properties of noise such as stationarity or Gaussianity. These approaches have shown great success in isolating signal components from noise and have been widely applied across various scenarios [182], [184]–[190].

Deep learning, on the other hand, offers a data-driven approach to denoising. Instead of relying on predefined statistical models, deep learning techniques learn directly from data in an attempt to capture the complex, non-linear relationships between noise and signal. This flexibility allows deep learning models to adapt to diverse and dynamic noise environments, such as those encountered in underwater acoustics. For example, with sufficient training data, these models could theoretically learn to suppress characteristic noise patterns like those of snapping shrimp or low-frequency ship hums.

The application of deep learning in underwater acoustic denoising is still in its early stages and comes with its own challenges. A key limitation is the scarcity of clean, noise-free reference data for supervised learning. Due to the intrinsic nature of the underwater environment, it is nearly impossible to obtain uncontaminated recordings of target signals underwater, complicating the generation of accurate ground truth labels. Additionally, the lack of large, diverse datasets representing various noise scenarios and target signals further constrains the effectiveness of deep learning approaches, which is highly dependent on the quality and quantity of training data. Finally, deep learning models require significant computational resources and expertise, which can be a barrier for their adoption in resource-constrained settings.

While deep learning does not address all the challenges inherent in underwater acoustic

denoising, its ability to model complex, non-linear, and context-dependent noise makes it a promising area for exploration. This chapter investigates the application of deep learning techniques to evaluate their effectiveness and potential in this domain.

6.2 Overview of denoising techniques

The deep learning revolution has introduced powerful methods for image denoising, paving the way for advancements in challenging domains like underwater acoustics [183], [191]. Denoising techniques can broadly be divided into mapping-based approaches, which directly transform noisy data into clean approximations, and masking-based approaches, which isolate certain regions of interest. In this section, we provide an overview of these techniques and their specific relevance to underwater acoustics. Understanding these methods will be essential for interpreting the experiments discussed later in this chapter, where we apply and assess their effectiveness in handling real-world underwater noise.

6.2.1 Mapping-based techniques

Mapping-based denoising techniques aim to directly transform a noisy input, such as a spectrogram or image, into a clean output, and hence are usually evaluated using image-comparison metrics like peak signal-to-noise ratio (PSNR) [192], [193]. These methods fall into two main categories: supervised and unsupervised. Supervised approaches require paired clean-noisy data, making them effective in controlled settings but less applicable in real-world scenarios where clean data is often unavailable. Unsupervised methods, such as Noise2Noise, overcome this limitation by learning directly from noisy data, leveraging noise properties or signal structure to achieve denoising without clean labels. The following subsections explore these techniques and their relevance to underwater acoustics.

Supervised methods

Supervised denoising methods rely on paired clean and noisy data for training, where each noisy input x is matched to its corresponding clean target y. The goal of these methods is to learn a mapping function f which, given a noisy input, outputs a denoised approximation $\hat{y}$ of the clean data:

$$\hat{y} = f(x) \qquad \text{where} \qquad \hat{y} \approx y \tag{6.3}$$

The training objective involves *minimising the empirical risk* over the paired clean-noisy dataset. More formally, the model parameters θ are optimised by minimising a chosen loss function L, such as mean squared error (L_2 loss), over all training pairs:

$$\arg\min_{\theta} \sum_{i} L(f_\theta(x_i), y_i) \tag{6.4}$$

Here, x_i is the noisy input, y_i is the corresponding clean target, and $\hat{y}_i = f_\theta(x_i)$ is the model's denoised output.

These experiments often begin by taking a clean image y and transforming it into a noisy version x using a noise model ϕ. For example, if the noise is Gaussian, the noisy image is generated as:

$$x = \phi(y) = y + \mathcal{N}(\mu, \sigma^2) \tag{6.5}$$

The model is then trained using x and y as pairs and the output $\hat{y}$ is compared to y using metrics such as PSNR.

The effectiveness of supervised methods hinges on the availability of clean-noisy paired datasets. While such data can be synthesised in controlled environments, this approach faces significant limitations when applied to real-world scenarios, particularly in the underwater domain where clearn data is impossible to obtain. Furthermor, when clean-noisy pairs are artificially generated by adding synthetic noise to clean data, the resulting noise distribution may not accurately represent real-world conditions. In underwater acoustics, noise arises from a range of sources, including equipment artifacts, environmental factors, and marine life, which are challenging to replicate using simple noise models.

Unsupervised methods

Unsupervised denoising methods address scenarios where clean data is unavailable. Instead of relying on explicit pairs of clean and noisy data, these methods leverage properties of the noise distributions or the inherent structure of the signal itself to perform denoising. This is particularly valuable in fields like underwater acoustics, where acquiring clean datasets is often infeasible.

One of the most influencial pieces of work in this domain is Noise2Noise (N2N), introduced by Lehtinen et al. in their seminal 2018 paper *Noise2Noise: Learning Image Restoration without Clean Data* [194]. The key discovery of Noise2Noise is that clean data is not strictly necessary for training denoising models. Instead, networks can learn to denoise using only pairs of independently corrupted noisy images, provided two critical assumptions are met:

1. Zero-mean noise: The noise must have a zero-mean distribution, ensuring that its expectation over the data cancels out. This allows the network to converge on the clean signal during training.

2. Uncorrelated noise: The noise in the input and target images must be uncorrelated (or ideally independent). This ensures the network does not inadvertently learn to map one type of noise to another, but rather focuses on extracting the underlying clean signal.

These assumptions are rooted in statistical properties. For example, under an L_2 loss function, the network's output converges to the arithmetic mean of the target distribution. For zero-mean noise, this means the network learns to approximate the clean signal as the noise cancels out on average.

Given these assumptions, the Noise2Noise training objective becomes:

$$\arg\min_{\theta} \sum_{i} L(f_\theta(x_i), x'_i) \tag{6.6}$$

where x_i and x'_i are two independent noisy realisations of the same underlying clean signal y_i.

Noise2Noise has inspired numerous adaptations in domains such as audio and underwater acoustics, where the availability of clean data is a persistent challenge. In the next section, we review three papers that extend the Noise2Noise framework to these fields.

Alamdari et al. (2020) Alamdari et al. [193] were the first to extend the Noise2Noise methodology to the audio domain, focusing on speech denoising. To address the challenge of limited clean data in real-world audio environments, the authors proposed a hybrid two-stage framework *Noisy2Noisy*, combining both supervised and self-supervised learning. In the first stage, the network was pre-trained on publicly available datasets with paired clean and noisy speech data. In the second stage, the network was fine-tuned using only noisy speech data, following the N2N principle.

To satisfy N2N assumptions, the authors assumed the noise to be zero-mean and employed a mid-side microphone setup to ensure noise decorrelation. The mid microphone captured the primary signal, while the side microphone captured ambient noise at a 90-degree angle. The authors argued that this configuration ensures decorrelation of noise between the channels due to differences in phase and magnitude.

While this approach demonstrates significant innovation, the reliance on clean data during pretraining partially undercuts its claim of self-supervised learning. Moreover, the assertion of noise decorrelation via the mid-side microphone has been debated [195]. Despite these limitations, Alamdari et al.'s work represents a key milestone, successfully adapting N2N to the audio domain and laying the groundwork for subsequent research.

Koh et al. (2020) Koh et al. [195] were the first to adapt the N2N methodology to the underwater acoustic domain through their WaveN2N approach. To address the core assumption – that noise is uncorrelated while the signal remains correlated – the authors used data from hydrophone arrays. These arrays consist of multiple spatially separated hydrophones, each capturing the same acoustic signal but with different realisations of noise. This spatial separation ensured noise decorrelation while maintaining signal consistency across the array, with only minor phase and time shifts caused by the slow-moving source.

The study provided qualitative results, showcasing spectrograms before and after denoising, where post-denoising signals revealed clearer structure and reduced noise levels. However, quantitative metrics such as PSNR or SNR improvement were absent, leaving the evaluation subjective and limited to visual analysis. Furthermore, the method relied on array recordings to meet the assumptions, which, as later noted by Zhou et al. [192], may limit its applicability in scenarios where only single-channel hydrophones or tightly spaced

arrays are available. However, the study still marks an important step towards adapting self-supervised denoising methods to underwater applications.

Zhou et al. (2023) Zhou et al. [192] adapted the Noise2Noise methodology to suppress self-noise in autonomous underwater vehicles, addressing challenges posed by mechanical, propeller, and hydrodynamic noise that often obscure target signals.

The authors employed the U-Net architecture with the log power spectrum used as both input and output features. To satisfy assumptions, random zero-mean and uncorrelated sinusoidal signals were added to noisy hydrophone data during training. At test time, the model estimated and subtracted the noise spectrum to produce a clean signal.

The approach was evaluated on single-channel data from 4-element and 16-element hydrophone arrays, achieving an average SNR improvement of 3.9 dB and a maximum improvement of 8.4 dB for the 16-element array.

Notably, Zhou et al.'s method represents a significant advancement by eliminating the need for multichannel data during training, unlike previous approaches such as Koh et al. [195]. This innovation reduces the cost and complexity of applying deep learning to underwater noise suppression and makes the approach more practical for real-world applications. While the reliance on synthetic signal additions may limit generalisation to other scenarios, the study represents a significant milestone in applying for single-channel noise suppression in underwater acoustics.

Together, these studies highlight the growing versatility of in audio and underwater acoustics, while also revealing key challenges, such as reliance on specific hardware setups or synthetic data, that must be addressed to enable broader applications.

Evaluation metrics

Peak signal-to-noise ratio The primary evaluation metric used in this chapter to quantify the similarity between two images is the peak signal-to-noise ratio (PSNR), which is widely used in denoising tasks to measure how closely the denoised version $\hat{y}$ resembles the original image y:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{6.7}$$

Here, MAX is the maximum possible pixel value in the image, and MSE is the mean squared error between y and $\hat{y}$:

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (y(i,j) - \hat{y}(i,j))^2 \tag{6.8}$$

where m and n are the dimensions of the image, and $y(i,j)$ and $\hat{y}(i,j)$ are the pixel values of the original and denoised images at position (i,j) respectively. PSNR essentially measures the ratio of the maximum possible power of a signal (MAX^2) to the power of the noise

(MSE). A high PSNR (above 30 dB) indicates that $\hat{y}$ is very similar to y. It is expressed in decibels.

Structural similarity index measure The structural similarity index measure (SSIM) is another widely used evaluation metric that quantifies the similarity between two images by comparing their structural information, luminance, and texture. Unlike PSNR, which primarily measures pixel-wise differences, SSIM takes into account the perceptual quality of images by evaluating how well the structures in the image are preserved during processing. It ranges from -1 to 1, where a value of 1 indicates perfect similarity between the two images, while a value of 0 or negative indicates a significant difference. SSIM is particularly useful in denoising and image restoration tasks, as it reflects not only pixel accuracy but also how well important visual structures are maintained, making it more aligned with human visual perception.

6.2.2 Masking-based techniques

Masking-based denoising techniques aim to isolate regions of interest in an image or spectrogram while suppressing irrelevant or noisy regions [192], [193]. Originating from image segmentation in computer vision, these methods generate binary or soft masks to highlight specific features while discarding unwanted areas. These masks can then be used for downstream tasks such as classification, making it a more targeted and extreme form of denoising.

Although masking techniques are well-established in other domains [196], their direct application to underwater acoustics remains relatively unexplored. This is partly due to the unique challenges posed by underwater environments. Spectrograms from acoustic signals often lack clear separations between signal and noise, complicating the generation of accurate masks. Additionally, the variability of underwater noise sources and conditions makes it difficult to generalise masking techniques across different datasets or applications.

In this chapter, we investigate masking-based techniques by applying principles of image segmentation to underwater acoustic spectrograms. By generating binary masks to isolate narrowband events, we aim to assess their utility for signal enhancement and classification. This work represents an initial step in adapting segmentation methodologies to the underwater domain and explores the potential of masking as a preprocessing tool for acoustic data.

Evaluation metrics

Binary accuracy Binary accuracy measures the percentage of correctly predicted pixels in a binary mask, where each pixel is classified as either belonging to the object of interest (foreground) or not (background). Mathematically, binary accuracy is computed as the

ratio of the number of correct predictions to the total number of predictions:

$$\text{Binary accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i) \tag{6.9}$$

where N is the total number of pixels, $\hat{y}_i$ is the predicted binary value, y_i is the ground truth binary value, and $\mathbb{1}$ is the indicator function which equals 1 if the prediction matches the ground truth and 0 otherwise. A higher binary accuracy indicates that the predicted regions in the mask closely match the ground truth, reflecting a higher proportion of true positives.

Intersection over union The intersection over union (IoU) metric measures the overlap between the predicted and ground truth masks, providing a more holistic evaluation of segmentation accuracy than binary accuracy. Binary IoU is a specific case of IoU when there are only two classes, 0 and 1. IoU is calculated as the ratio of the intersection of the predicted and ground truth masks to their union:

$$\text{IoU} = \frac{|\hat{y} \cap y|}{|\hat{y} \cup y|} \tag{6.10}$$

where $\hat{y}$ represents the predicted binary mask, y represents the ground truth binary mask, and the union and intersection are calculated pixel-wise. A higher IoU indicates a better alignment between the predicted and actual regions of interest. Since IoU penalises false positives and false negatives, it provides a more informative measure of segmentation quality compared to binary accuracy, especially in complex or noisy images.

6.3 Overview of model architectures explored

Given the constraints and challenges that come with this line of research, encoder-decoder architectures – specifically, *autoencoders*, as we will be training in an unsupervised manner – naturally emerge as the most suitable framework for our denoising task. These structures encode noisy inputs into a latent representation, allowing the model to focus on essential features, before decoding the representation back into a denoised spectrogram (Section 2.3.3).

This chapter examines two encoder-decoder architectures: one simple and one more complex, to enable a comparative analysis between the two. The details of their structures are outlined below.

6.3.1 Irfan

The most basic model employed in this chapter is a straightforward convolutional encoder-decoder architecture, adapted from Irfan et al.'s 2020 paper, *A Novel Feature Extraction Model to Enhance Underwater Image Classification* [197]. Originally developed for underwater image classification tasks using datasets like Fish4Knowledge and ImageNet un-

derwater synsets, the model demonstrated strong performance in improving classification results over established architectures such as Inception and DenseNet. The decision to use this model comes from its simplicity, clarity in design, and well-structured explanations provided in the source paper. It also served as an excellent starting point for learning encoder-decoder models, making it an ideal baseline for the denoising experiments in this chapter. For brevity, this architecture will hereafter simply be referred to as *Irfan* or *the Irfan model*.

Irfan is a convolutional encoder-decoder structure. The encoder progressively reduces the spatial dimensions of the input while capturing essential features through convolutional and pooling operations. The decoder then reconstructs the denoised signal, utilising upsampling layers and transpose convolutions to restore the original resolution. A final sigmoid activation ensures that the output values are normalised between 0 and 1, aligning with the intensity values of the spectrogram data.

The original architecture included a softmax classification layer in the bottleneck, designed for supervised classification tasks. For our unsupervised denoising objective, this layer was removed, resulting in a fully sequential architecture focused solely on reconstructing clean spectrograms. Additional modifications were made to adjust the size of the filters and the number of convolutional blocks to better suit the requirements of our task.

The architectural details are summarised in Table 6.1 and illustrated in Figure 6.1.

6.3.2 U-Net

The U-Net model, originally proposed by Ronneberger et al. in their seminal 2015 paper, *U-Net: Convolutional Networks for Biomedical Image Segmentation* [106], is an extremely successful and well-recognised benchmark encoder-decoder architecture. Originally developed for image segmentation tasks, U-Net is characterised by its symmetric design: an encoder path that progressively reduces spatial dimensions to capture contextual information, and a decoder path that reconstructs high-resolution outputs via up-sampling and skip connections. Its name comes from the U-shaped structure formed by the encoder, decoder, and the connecting skip connections.

Skip connections, first popularised by the ResNet architecture [87], are one of the key features in U-Net. They help gradients flow smoothly from the input to the output layers, making training more stable and addressing issues like the vanishing gradient problem [198]. Additionally, these connections carry over spatial details lost during down-sampling in the encoder and combine them with features in the decoder. This ensures that the model reuses important details from earlier layers and maintains consistent dimensions, helping to merge fine-grained spatial information with the broader context captured by earlier layers.

In this study, the classification head of the original U-Net has been removed to focus exclusively on reconstruction tasks. The architectural details are summarised in Table 6.2, and the model is illustrated in Figure 6.2.

Table 6.1: Layer-wise breakdown of the modified encoder-decoder architecture inspired by Irfan et al. [197]. Convolutional and transpose convolutional layers include batch normalisation and ReLU activation (not explicitly shown).

Block	Layer Name	Filter Size	Output Shape
Encoder	Input	–	$192 \times 192 \times 1$
	Conv1	3×3	$192 \times 192 \times 64$
	MaxPool1	2×2	$96 \times 96 \times 64$
	Conv2	3×3	$96 \times 96 \times 64$
	MaxPool2	2×2	$48 \times 48 \times 64$
	Conv3	3×3	$48 \times 48 \times 128$
	MaxPool3	2×2	$24 \times 24 \times 128$
	Conv4	3×3	$24 \times 24 \times 256$
	MaxPool4	2×2	$12 \times 12 \times 256$
Decoder	ConvTrans1	3×3	$12 \times 12 \times 256$
	UpSample1	2×2	$24 \times 24 \times 256$
	ConvTrans2	3×3	$24 \times 24 \times 128$
	UpSample2	2×2	$48 \times 48 \times 128$
	ConvTrans3	3×3	$48 \times 48 \times 64$
	UpSample3	2×2	$96 \times 96 \times 64$
	ConvTrans4	3×3	$96 \times 96 \times 64$
	UpSample4	2×2	$192 \times 192 \times 64$
	ConvTrans5	3×3	$192 \times 192 \times 1$
	Output (Sigmoid)	–	$192 \times 192 \times 1$

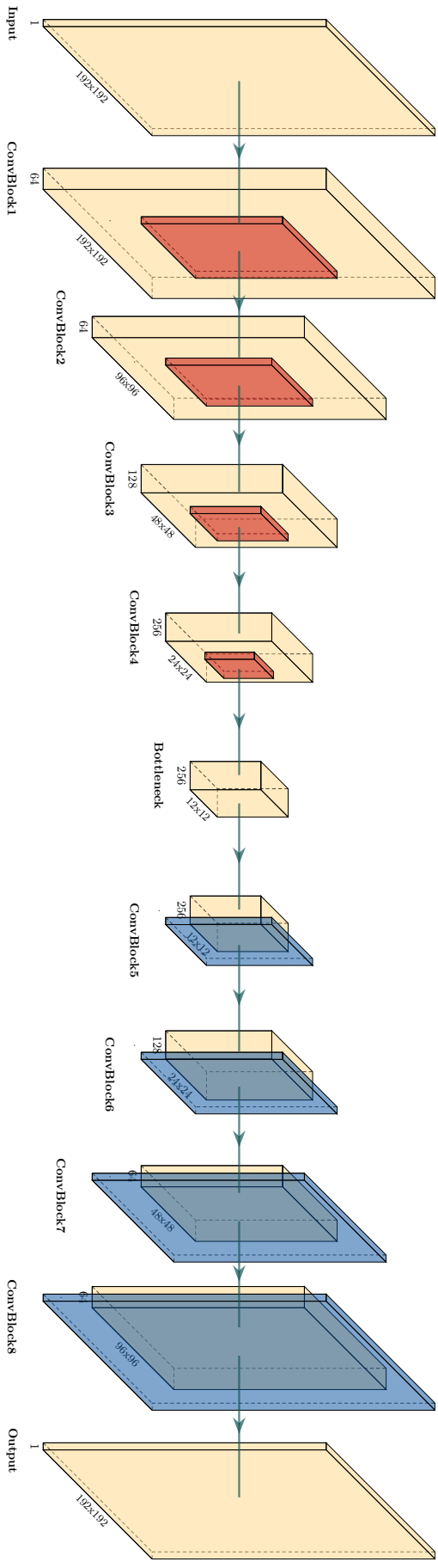


Figure 6.1: Modified convolutional encoder-decoder architecture inspired by Irfan et al. (2020) [197]. Each ConvBlock includes batch normalisation and ReLU activation, though these are not explicitly shown.

Table 6.2: Layer-wise breakdown of the U-Net architecture. Convolutional layers are each followed by LeakyReLU activation (not explicitly shown).

Block	Layer Name	Filter Size	Output Shape	Connected to
Encoder	Input	–	$192 \times 192 \times 3$	–
	Conv1	3×3	$192 \times 192 \times 64$	Input
	Conv2	3×3	$192 \times 192 \times 64$	Conv1
	MaxPool1	2×2	$96 \times 96 \times 64$	Conv2
	Conv3	3×3	$96 \times 96 \times 128$	MaxPool1
	Conv4	3×3	$96 \times 96 \times 128$	Conv3
	MaxPool2	2×2	$48 \times 48 \times 128$	Conv4
	Conv5	3×3	$48 \times 48 \times 256$	MaxPool2
	Conv6	3×3	$48 \times 48 \times 256$	Conv5
	MaxPool3	2×2	$24 \times 24 \times 256$	Conv6
	Conv7	3×3	$24 \times 24 \times 512$	MaxPool3
	Conv8	3×3	$24 \times 24 \times 512$	Conv7
	MaxPool4	2×2	$12 \times 12 \times 512$	Conv8
Bottleneck	Conv9	3×3	$12 \times 12 \times 1024$	MaxPool4
	Conv10	3×3	$12 \times 12 \times 1024$	Conv9
Decoder	UpSample1	2×2	$24 \times 24 \times 1024$	Conv10
	Concatenate1	–	$24 \times 24 \times 1536$	UpSample1, Conv8
	Conv11	3×3	$24 \times 24 \times 512$	Concatenate1
	Conv12	3×3	$24 \times 24 \times 512$	Conv11
	UpSample2	2×2	$48 \times 48 \times 512$	Conv12
	Concatenate2	–	$48 \times 48 \times 768$	UpSample2, Conv6
	Conv13	3×3	$48 \times 48 \times 256$	Concatenate2
	Conv14	3×3	$48 \times 48 \times 256$	Conv13
	UpSample3	2×2	$96 \times 96 \times 256$	Conv14
	Concatenate3	–	$96 \times 96 \times 384$	UpSample3, Conv4
	Conv15	3×3	$96 \times 96 \times 128$	Concatenate3
	Conv16	3×3	$96 \times 96 \times 128$	Conv15
	UpSample4	2×2	$192 \times 192 \times 128$	Conv16
	Concatenate4	–	$192 \times 192 \times 192$	UpSample4, Conv2
	Conv17	3×3	$192 \times 192 \times 64$	Concatenate4
	Conv18	1×1	$192 \times 192 \times 64$	Conv17
	Output	1×1	$192 \times 192 \times 3$	Conv18

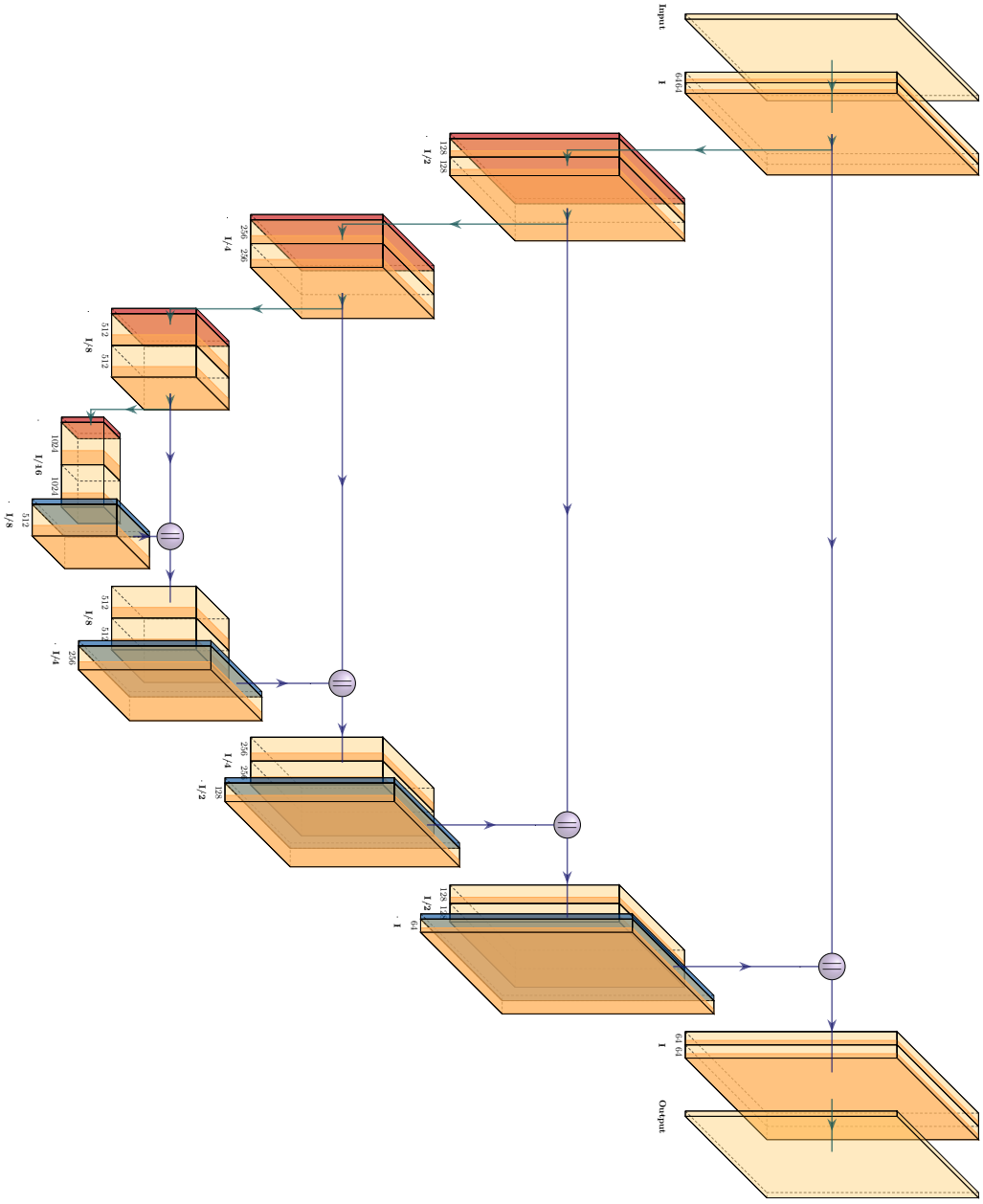


Figure 6.2: Illustration of the U-Net architecture [106] with its classification layer removed, resulting in a fully symmetric encoder-decoder structure.

6.4 Overview of benchmark datasets used

To evaluate the performance of our denoising methods, we used two well-established natural image datasets as benchmarks: ImageNet and the Berkeley Segmentation Dataset (BSD).

6.4.1 ImageNet

ImageNet [81] is a landmark dataset widely used for training and benchmarking computer vision models. It consists of over 15 million high-resolution images categorised into approximately 22,000 classes, collected from web sources such as Flickr. Since 2010, the dataset has served as the foundation for the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [199], which has been pivotal in driving breakthroughs in deep learning architectures, such as AlexNet [85], ResNet [87], and Vision Transformers [91].

For this chapter, we utilised the ILSVRC 2012 dataset, a widely used subset of ImageNet consisting of 1.2 million training images, 50,000 validation images, and 150,000 test images, evenly distributed across 1,000 object categories. Due to computational constraints, we worked with a subset of the validation set, specifically the first 10,000 images. This reduced dataset provided a practical sample for our experiments.

6.4.2 Berkeley Segmentation Dataset

The Berkeley Segmentation Dataset [200] is a widely used benchmark dataset in computer vision, particularly for tasks involving image segmentation and denoising. Unlike datasets designed for model training, BSD is primarily used for evaluating the performance of algorithms across metrics such as PSNR. The dataset consists of 300 natural images, divided into 200 training images and 100 testing images. The images are sourced from everyday scenes, including landscapes, objects, and urban settings, and have been manually annotated to facilitate segmentation tasks. For this chapter, we used the test set of 100 images for the evaluation of our methods on natural images.

6.5 Experiment 1: Unsupervised denoising with Noise2Noise

The Noise2Noise framework offers an innovative approach to denoising, bypassing the need for clean labels by leveraging pairs of noisy samples with uncorrelated noise. As outlined in Section 6.2, this methodology has demonstrated remarkable success in domains where these assumptions hold true. The objective of this experiment is twofold: first, to validate the Noise2Noise approach by recreating its performance on natural images; and second, to investigate its applicability to underwater acoustic spectrograms, a domain where these assumptions are challenging to approximate.

This experiment is structured into two parts. In the first part, we replicate the Noise2Noise framework using natural images, employing two models, Irfan and U-Net, to benchmark their performance. This serves as a baseline validation of the methodology, and verification of the results originally presented in the Noise2Noise paper. In the

second part, we extend the framework to underwater acoustic spectrograms, testing its performance under the specific assumptions and constraints of this domain. This section examines whether Noise2Noise can effectively handle the complexities of underwater acoustics, including dynamic noise conditions and the absence of clean paired data.

Ultimately, this experiment seeks to provide insights into the strengths and limitations of Noise2Noise, offering a foundation for its future application to underwater acoustic signal processing. We conclude by reflecting on the challenges encountered and proposing directions for further research.

6.5.1 Part I: Recreating the original paper

This section aims to validate the performance of the Noise2Noise methodology on natural images by recreating the original paper as authentically as possible using our two chosen architectures: U-Net and the Irfan model. By replicating the Gaussian noise experiments from the original Noise2Noise paper [194], we demonstrate the viability of the approach before applying it to underwater acoustic spectrograms.

Methodology

Our methodology closely follows the original Noise2Noise framework, with some modifications to accommodate our experimental setup and computational constraints. Note that the original paper compares their technique traditional supervised denoising; thus, we also implement a traditional denoiser using both Irfan and U-Net for comparison.

Training Noise2Noise In line with the original paper, we began by taking clean images from the ImageNet validation set and extracting a random square patch of 192×192 pixels from the image. Each patch was then processed to generate two distinct noisy versions by applying Gaussian noise with a standard deviation randomly sampled from the range $[0, 50]$, as specified in the original paper. This process created noisy image pairs (x, x'), with one noisy image x serving as the model input, while the other x' acting as the target label. This setup ensured compliance with the Noise2Noise assumption that paired noisy inputs share the same underlying distribution.

Training a supervised denoiser For comparison, we also trained a traditional supervised denoiser on the ImageNet validation set. In this case, a random 192×192 pixel patch was extracted from each image, but instead of applying two different noise distributions, only one was applied. This process created noisy-clean pairs (y, x).

At the end of each epoch, the performance of all models was evaluated using a validation set consisting of the BSD300 training images. The validation generator produced pairs (y, x), allowing us to compute the PSNR by directly comparing the denoised output $\hat{y}$ with the ground truth, y.

Finally, after training, each model was evaluated on the BSD test set. As before, a random 192×192 pixel crop was extracted from each test image, and Gaussian noise with a random standard deviation was applied. The noisy patch was fed into the model for evaluation, with the denoised outputs compared to the clean originals. The PSNR values were recorded to assess the effectiveness of Noise2Noise on natural images and to provide a direct performance comparison between the U-Net and Irfan architectures.

Implementation and training setup

We implemented the training and validation pipelines in Python using TensorFlow. The implementation was designed with flexibility in mind, offering options to toggle 0-1 normalised inputs, convert images to greyscale or retain RGB channels, and switch between multiple noise distributions if required. For 0-1 normalised inputs, the standard deviation of the Gaussian noise was scaled by a factor of 255 to match the scaled pixel intensity range.

Our training setup aimed to follow the original paper as faithfully as possible: the minibatch size was set to 4 for training and evaluation, training was conducted using the ADAM optimiser [162], with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$, mean squared error (MSE) was used as the loss function, and PSNR served as the primary evaluation metric.

For the U-Net model, weights were initialised using He initialisation [201] for stable convergence. No batch normalization, dropout, or other regularisation techniques were used, following the original methodology. Finally, LeakyReLU with $\alpha = 0.1$ [202] was employed after each convolutional layer except the final one.

Our complete implementation of Noise2Noise can be found on the project GitHub repository.

Adjustments to original settings

While our goal was to faithfully recreate the Noise2Noise paper, several adjustments were made to align with our specific experimental setup and resource constraints:

- The original paper explored multiple including Poisson, and Bernoulli distributions. However, as our aim here was primarily verification of the method, we decided to focus exclusively on Gaussian noise for simplicity and computational efficiency.

- In the original paper, the authors conducted some of their initial experiments using an architecture called RED30 [203] but quickly switched to U-Net as it was "roughly 10x faster" while still giving "similar results" [106, p. 3]. Here, we simply used the U-Net and Irfan architectures for our experiment.

- Due to computational limitations, we used a subset of 10,000 images from the ImageNet validation set instead of the full 50,000 images.

- For similar reasons, we also limited training for the U-Net model to 50 epochs compared to the original 150 epochs [194, Fig 1.].

- The image patches extracted from the images were 192×192 pixels rather than the 256×256 stated in the original implementation, in order to ensure consistency with the resolution of DeepShip spectrograms.
- Our experiments were conducted on grayscale images (1 channel), reflecting the single-channel nature of spectrograms, whereas the original paper worked with RGB images (3 channels).
- The paper used $[-0.5, 0.5]$ normalisation, while we chose to use $[0, 1]$ normalisation as it is a more standard technique including when dealing with spectrograms.
- The original paper used a learning rate of 10^{-3}, but preliminary experiments showed suboptimal results with this setting. We adjusted this to 10^{-5} for more stable training.
- Unlike the original, which did not specify a validation set, we introduced a validation step using BSD300's training set which contains 200 images. Nevertheless, evaluation was still conducted on BSD300's testing set of 100 images, as the original paper called for.

Results

The results of our recreation of the Noise2Noise paper closely align with those obtained by the original authors. The Irfan model achieved a PSNR of 22.15 dB on the BSD testing set, while the U-Net model, a deeper and more sophisticated architecture, achieved 29.59 dB. This result is slightly below the 31.06 dB obtained by Lehtinen et al. [194, p. 3], who trained their model for 100 more epochs than we did. A visualisation of the Noise2Noise model outputs is presented in Figure 6.3. Additionally, Figure 6.4 compares the outputs from supervised and unsupervised models.

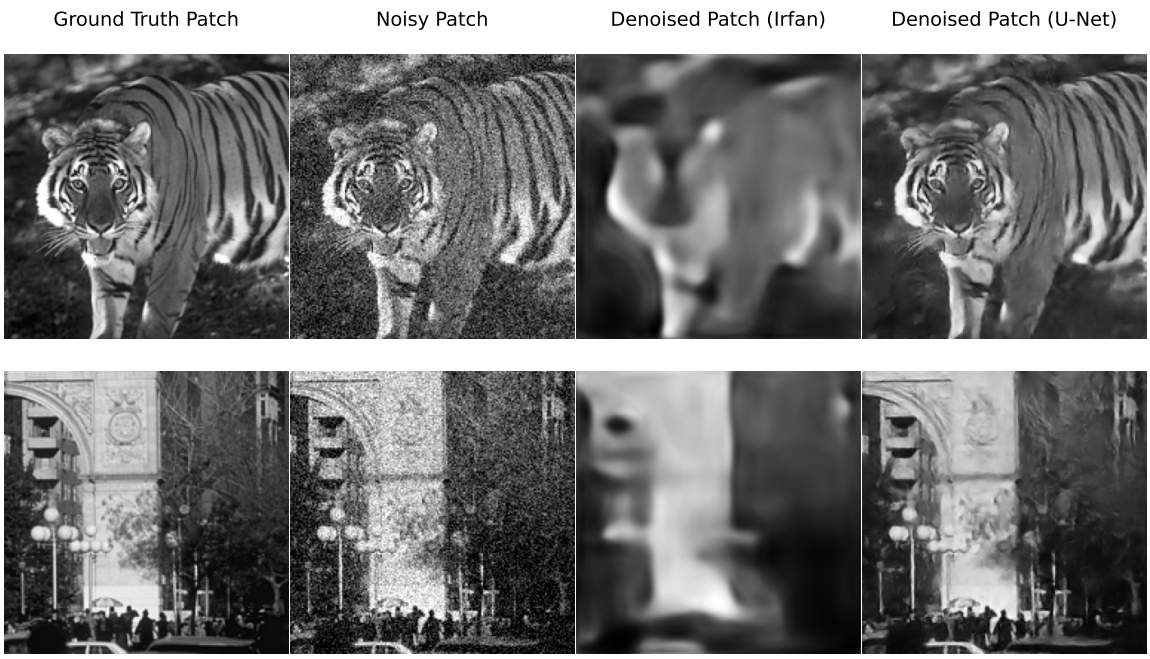
Table 6.3: Performance of the Irfan and U-Net models on the BSD testing set for our recreation of the Noise2Noise paper.

Strategy	Model	Loss	PSNR (dB)
Supervised	Irfan	0.0094	21.48
	U-Net	0.0017	29.78
Noise2Noise	Irfan	0.0080	22.15
	U-Net	00.18	29.59

Discussion

The results of this experiment validate the effectiveness of the Noise2Noise methodology, with the unsupervised model achieving performance nearly identical to that of the supervised model in terms of PSNR. This outcome is consistent with the findings of the original Noise2Noise paper [194]. Additionally, the outputs from both the supervised and unsupervised models (Figure 6.4) appear nearly identical, reinforcing the idea that the Noise2Noise

(a) Raw ground truth images used for visual evaluation with 192×192 patches highlighted.



(b) Comparison of ground truth patches, noisy patches, the Irfan model output, and the U-Net model output.

Figure 6.3: Visual comparison of ground truth images, noisy patches, and denoised model outputs for our Noise2Noise recreation.

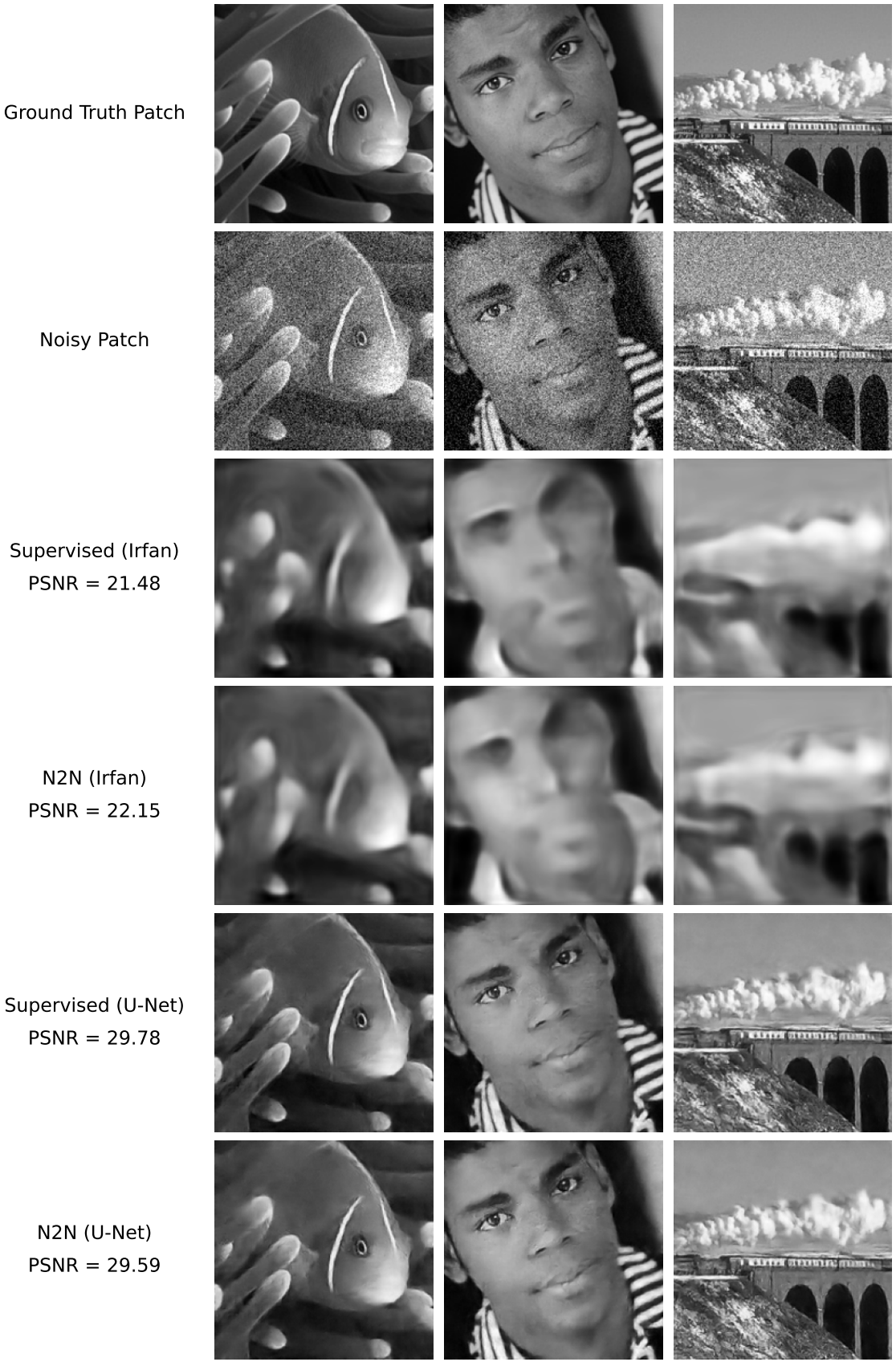


Figure 6.4: Comparison of supervised and Noise2Noise denoising methods applied to both the Irfan and U-Net models.

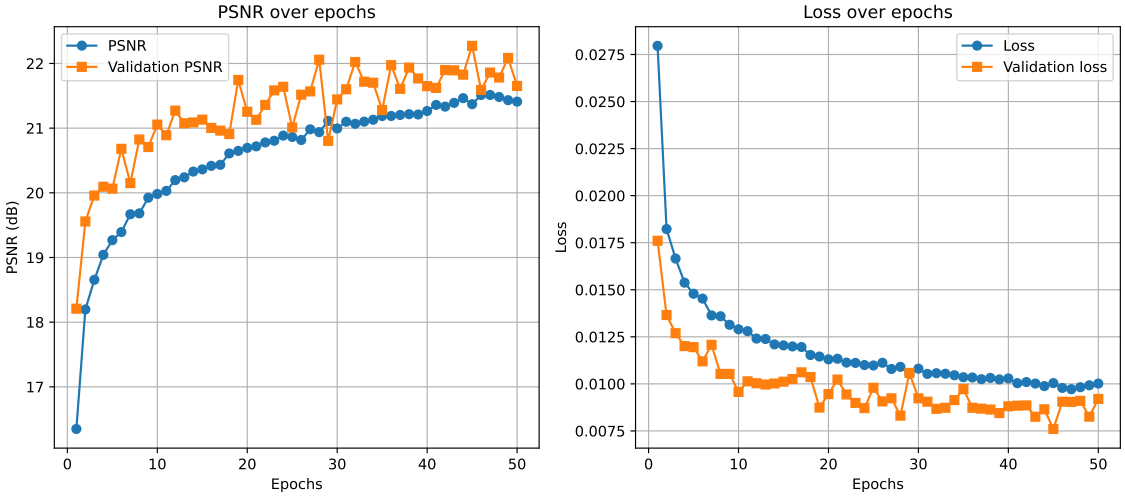
approach performs on par with supervised learning techniques in terms of visual quality. This highlights the groundbreaking nature of the Noise2Noise methodology, where noise reduction can be achieved without the need for clean target images, a challenge that has faced traditional supervised denoising techniques for decades.

Interestingly, the Irfan model trained with Noise2Noise slightly outperformed its supervised counterpart in PSNR and loss values. This minor discrepancy may stem from the inherent regularisation effect of Noise2Noise training, which uses noisy targets and potentially mitigates overfitting to specific clean patterns. However, the difference is small enough to conclude that both models perform similarly in practice, as further supported by the nearly identical visual outputs.

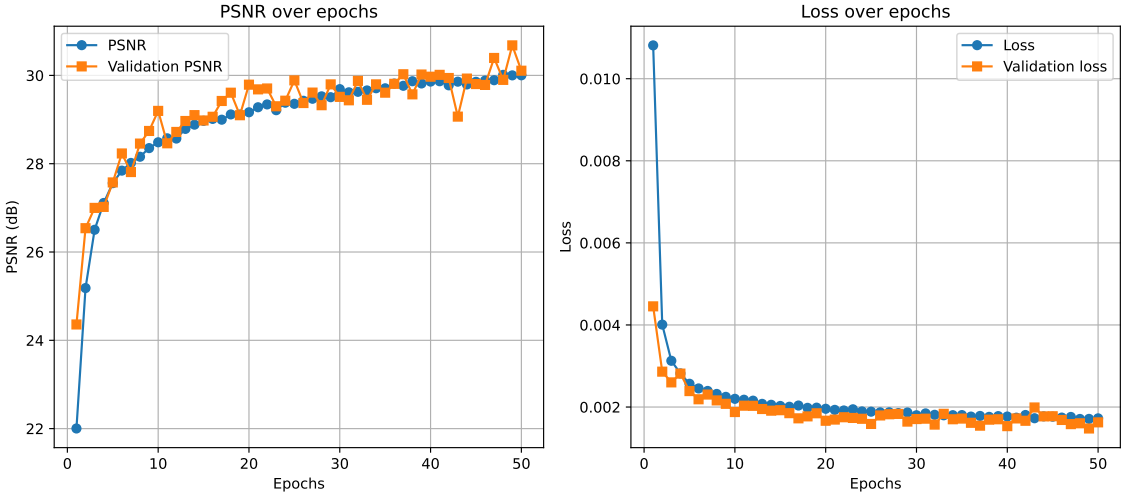
A comparison between the Irfan and U-Net architectures reveals more pronounced differences, particularly in their ability to capture fine details. U-Net consistently outperformed Irfan in reconstructing intricate features, such as the tiger's stripes (test/108005), the fine facial details of a man (test/302008), and the complex outlines and shading in the Washington Square Arch image (test/148089). In contrast, the Irfan model often produced blurry and less detailed outputs, failing to preserve critical image structures. Even in cases where both models succeeded at broader reconstructions, such as discerning the arches of the bridge in the steam train image (test/182053), U-Net demonstrated superior fidelity in capturing nuanced details like the shading of steam.

This performance discrepancy can be attributed to the architectural differences between Irfan and U-Net. Despite both Irfan and U-Net being convolutional encoder-decoder structures at heart, U-Net's success over Irfan is largely due to its skip connections, which allow the network to retain spatial information that would otherwise be lost during downsampling in the encoder. As discussed in Section 6.3.2, these connections help merge both high-level and low-level features, enabling U-Net to effectively reconstruct both macro and micro details in the images. On the other hand, Irfan, a purely sequential model, lacks this feature and thus cannot effectively capture fine-grained spatial details, leading to poorer performance in reconstructing complex image structures. This architectural limitation underscores why U-Net is widely considered a state-of-the-art model for image reconstruction tasks.

Examining the training curves (Figures 6.5 and 6.6) further supports these findings. All models demonstrated stable convergence, with PSNR values plateauing around the 50th epoch. While training to the originally intended 150 epochs would have been ideal, the computational constraints of this experiment limited training to 50 epochs, with each model requiring over 24 hours to train on the available hardware. The supervised model's training and validation curves were closely aligned, indicating a well-fitted model with minimal overfitting. Conversely, the Noise2Noise model showed slightly lower training PSNR compared to validation PSNR, which is consistent with the methodology's reliance on noisy targets for training.

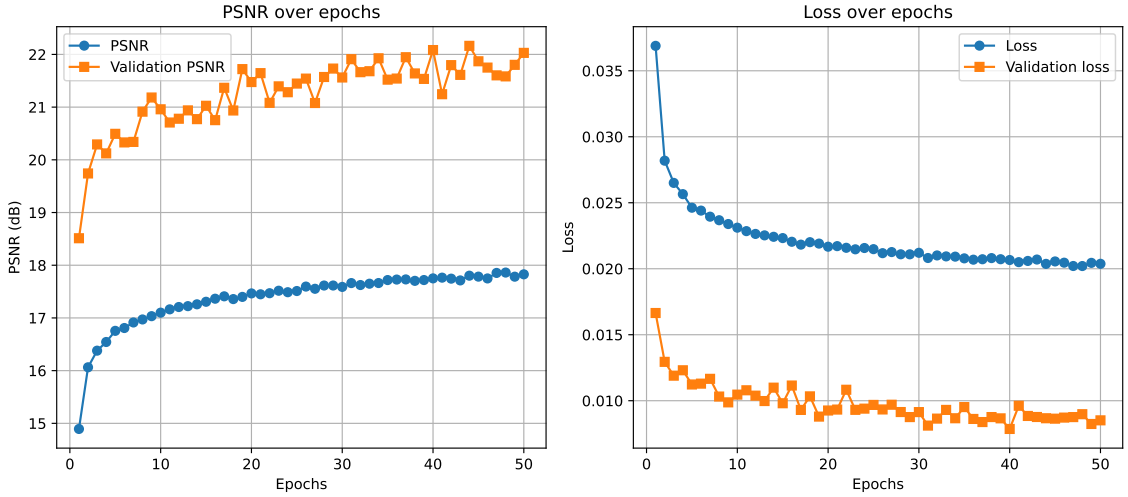


(a) PSNR and loss curves for Irfan (supervised denoising).

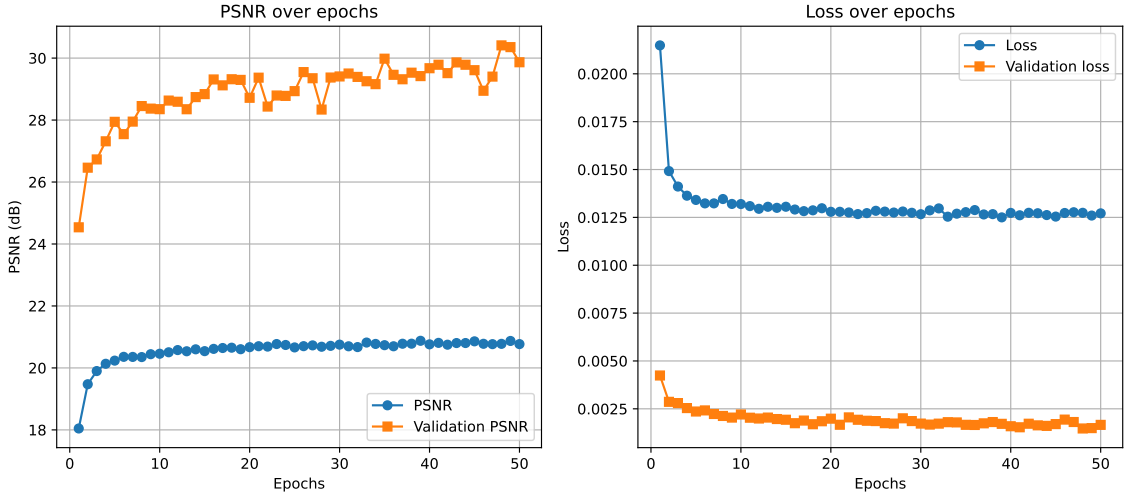


(b) PSNR and loss curves for U-Net (supervised denoising).

Figure 6.5: PSNR and loss curves for the supervised denoising models trained on (a) Irfan and (b) U-Net.



(a) PSNR and loss curves for Irfan (Noise2Noise denoising).



(b) PSNR and loss curves for U-Net (Noise2Noise denoising).

Figure 6.6: PSNR and loss curves for training the Noise2Noise model on (a) Irfan and (b) U-Net.

Conclusion

Overall, these results reaffirm the efficacy of the Noise2Noise methodology. The U-Net model, in particular, stands out for its ability to reconstruct fine image details, further validating its role as a robust solution for denoising tasks. Though these results set a strong foundation for applying Noise2Noise to spectrograms, challenges such as the absence of paired noisy spectrograms will require careful consideration.

6.5.2 Part II: Transitioning to underwater acoustic spectrograms

The previous section showed unequivocally that the Noise2Noise methodology works on natural images. We now turn our attention to the central question of this chapter: can Noise2Noise be effectively adapted to underwater acoustic spectrograms?

While the Noise2Noise methodology has demonstrated its robustness in domains with clearly defined noise characteristics, the transition to underwater acoustics presents unique challenges. The technique relies on two key assumptions: having 1. paired data of the same event with 2. uncorrelated, zero-mean noise. In underwater acoustics, fulfilling these requirements is far from straightforward. The lack of public hydrophone array datasets limits our access to multiple recordings of the same event, and underwater noise often deviates from the strict zero-mean condition.

The objective of this experiment is to explore whether these assumptions can be approximated. Instead of relying on hydrophone array recordings of the same event, we propose using spectrograms from the same vessel recorded at different times. By pairing spectrograms that share the same underlying signal but with varying noise, we aim to adapt the Noise2Noise framework to the complexities of underwater acoustic data. This experiment serves as an initial step in evaluating the feasibility of leveraging Noise2Noise for denoising tasks in this challenging and unexplored domain.

Methodology

To explore the application of Noise2Noise on underwater acoustic spectrograms, we began by identifying segments from the DeepShip dataset corresponding to vessels with multiple recordings. This step was necessary as the Noise2Noise framework relies on pairing spectrograms from the same vessel recorded at different times to approximate a shared underlying signal. Applying this filter reduced the dataset to 37,377 spectrograms.

The dataset was manually partitioned into training, validation, and testing sets in a 70-10-20 ratio. Care was taken to ensure that all recordings associated with a specific vessel were confined to the same split. This precaution was necessary to prevent scenarios where the algorithm would fail to find related spectrogram pairings for certain vessels, leading to errors during training.

We developed a custom data generator, `N2NDeepShipGenerator`, to streamline the pairing process. During initialisation, the generator precomputed all possible spectrogram pairings for efficiency. At runtime, it produced random pairs of spectrograms from the same

vessel but from different recordings. This pairing ensured that the noise was uncorrelated while the underlying signal remained similar, approximating the core assumptions of the Noise2Noise framework.

Evaluating the models posed a unique challenge due to the lack of a clean ground truth, which precluded the use of conventional metrics like PSNR. Instead, we employed two alternative metrics: mean squared error loss and SSIM. While MSE quantified the overall reconstruction error, SSIM offered insight into how well the structural content of the spectrograms was preserved.

Both the Irfan and U-Net models were trained for 50 epochs each. While training to 150 epochs was ideal (as in the original Noise2Noise paper), computational limitations necessitated a shorter training duration. The training pipeline was implemented using TensorFlow, and all experiments were conducted on an Alienware laptop equipped with GPU acceleration to manage the computational demands of the large dataset.

Results

The results of this experiment, summarised in Table 6.4, show the loss and SSIM metrics for the Irfan and U-Net models. Visual outputs from the models are presented in Figure 6.7. Unfortunately, the experiment yielded underwhelming results, with both models achieving extremely low SSIM scores, indicating poor structural similarity between the input and output spectrograms.

In addition to poor performance metrics, the models failed to converge meaningfully during training. As seen in Figure 6.8a, the Irfan model's validation loss showed no downward trend, suggesting that the model struggled to learn any effective mappings. While the U-Net model demonstrated some convergence (Figure 6.8b), the improvement in loss was minimal, decreasing by only 0.002 over 50 epochs. This lack of progress indicates significant challenges in adapting the Noise2Noise methodology to the underwater acoustic spectrogram dataset under the current experimental conditions.

Table 6.4: Comparison of loss and SSIM values for the Irfan and U-Net models under the Noise2Noise approximation.

Model	Loss	SSIM
Irfan	2.12×10^{-2}	0.118
U-Net	1.77×10^{-2}	0.129

Discussion

The visualised outputs (Figure 6.7) reveal the core problem: both the Irfan and U-Net models produced blurred and overly smoothed spectrograms, failing to preserve key features of the input. Despite the U-Net slightly outperforming the Irfan model in terms of loss and SSIM, neither model succeeded in creating outputs that could be considered meaningful or useful for downstream tasks.

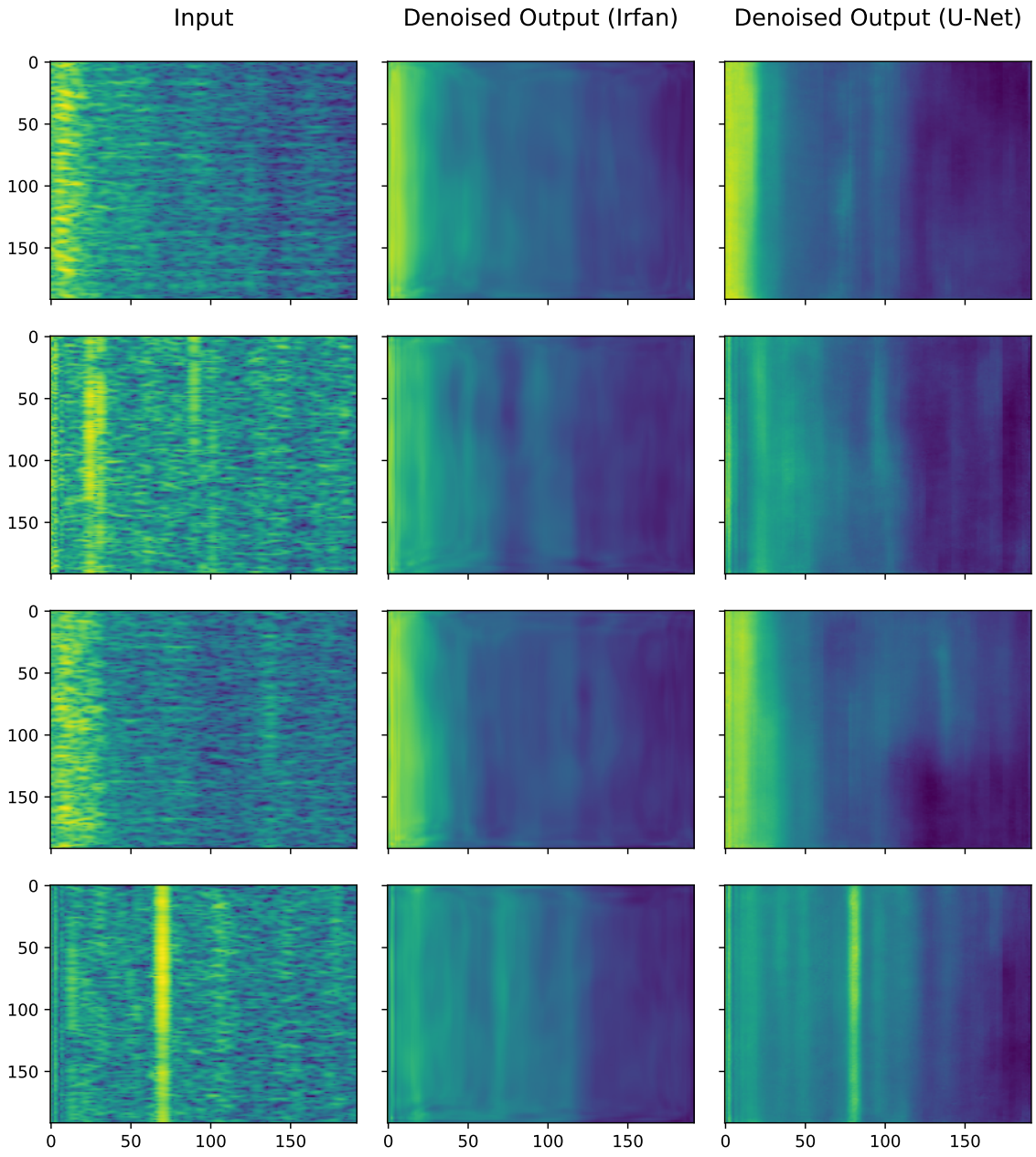
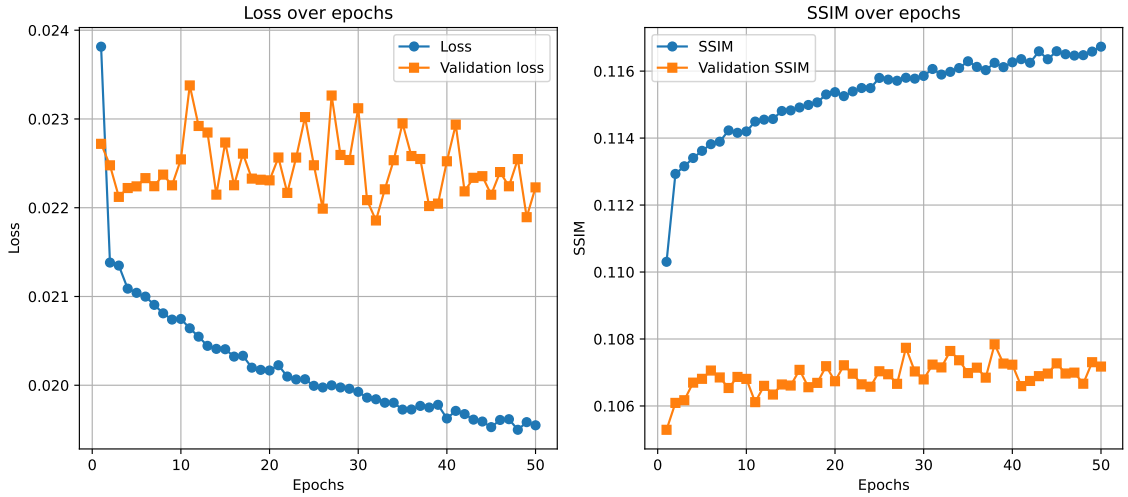
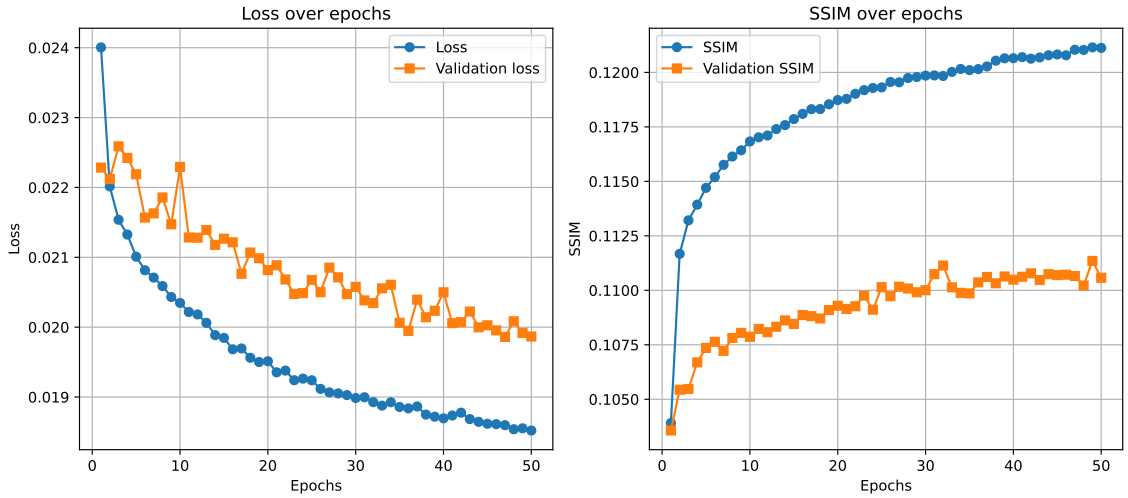


Figure 6.7: Outputs generated by the Irfan and U-Net models for the Noise2Noise experiment on spectrograms. The visualisations reveal heavily blurred and smoothed spectrograms that fail to retain key features from the original input.



(a) Loss and SSIM curves for the Irfan model.



(b) Loss and SSIM curves for the U-Net model.

Figure 6.8: Loss and SSIM curves for attempting to approximate Noise2Noise denoising principles on spectrogram data.

These results make it clear that the Noise2Noise framework is highly reliant on its assumptions: the input-output pairs must share the same underlying signal but have independent noise distributions. In the context of this experiment, these conditions were only approximated by pairing spectrograms from the same vessel recorded on different days. However, underwater noise is highly dynamic, influenced by environmental factors such as weather, water currents, and background biological activity. These factors likely introduced significant discrepancies in the noise distributions of the paired spectrograms, undermining the Noise2Noise methodology's core premise. Without truly independent noise distributions and a shared signal, the models struggled to identify and learn meaningful patterns.

The poor performance metrics and lack of convergence observed in this experiment strongly suggest that the current dataset and pairing strategy are inadequate for successfully applying Noise2Noise to underwater acoustic spectrograms. To address these challenges, future work should focus on improving pairing strategies, ideally through collecting hydrophone array data to provide multiple recordings of the same event with truly independent noise distributions. This would align more closely with the Noise2Noise framework.

Conclusion

This experiment highlights the significant challenges of adapting the Noise2Noise framework to underwater acoustic spectrograms. Despite its success in other domains, the methodology struggled under the constraints of this experiment, yielding outputs that failed to preserve meaningful spectrogram features. The core issue lies in the inability to satisfy the Noise2Noise assumptions: the paired spectrograms did not share identical underlying signals with independent noise distributions, as environmental factors such as weather and biological activity introduced substantial variability in the noise conditions between recordings.

Although the experiment was unsuccessful, it underscores the importance of developing more robust pairing strategies and dataset collection techniques, such as leveraging hydrophone arrays to capture multiple recordings of the same event with minimal noise variability. Ultimately, these insights provide a foundation for future research aimed at refining the application of Noise2Noise principles to underwater acoustic data.

6.6 Experiment 2: Masking-based denoising

This experiment evaluates the effectiveness of masking-based denoising for underwater acoustic spectrograms, specifically using the DeepShip dataset. The objective is to develop a deep learning model capable of accurately segmenting narrowband events from spectrograms while effectively removing noise. Additionally, the experiment explores the potential of automatic masking techniques to enhance the segmentation process.

6.6.1 Methodology

The development of an image segmentation model to extract narrowband events from spectrograms required both the original spectrogram images and their corresponding ground-truth masks. Ground-truth masks are essential for training as they highlight regions of interest within the spectrograms, providing the model with guidance on what to focus on during training.

The original spectrogram images were generated using the custom `wavToSpec()` function, described in Section 3.2, which includes the capability to export spectrograms in `.png` format.

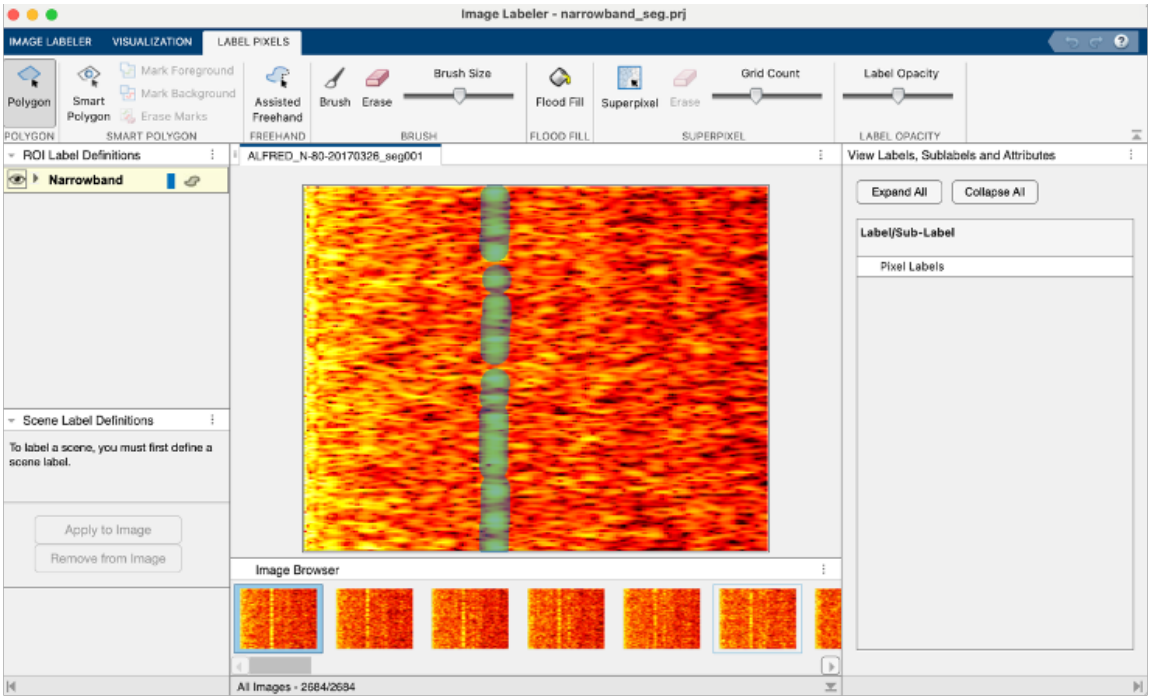
The primary challenge lay in the creation of ground-truth masks, as the DeepShip dataset does not include pre-existing labelled data. This limitation stems from the niche nature of the dataset and the field of underwater acoustic research, compounded by the labour-intensive and expertise-dependent nature of spectrogram annotation. Unlike conventional image labelling tasks, annotating spectrograms requires domain-specific knowledge to accurately identify features of interest, such as vessel-related narrowband tracks. Furthermore, the labelling process is sensitive to the specific spectrogram parameters and export configurations used, adding an additional layer of complexity. Consequently, the generation of ground-truth masks necessitated either manual labelling or the use of an automated masking algorithm.

Manual labelling was chosen for the purposes of this experiment. Using MATLAB's *Image Labeler* application, a total of 356 spectrograms were manually labelled to highlight narrowband frequencies and tracks of interest (Figure 6.9a). This process was quite labour-intensive, requiring careful annotation of each spectrogram. To facilitate model training, custom MATLAB functions were developed to convert the proprietary mask format produced by the *Image Labeler* into binary masks. In this format, the background is represented by pixel values of 0, while the regions of interest are represented by pixel values of 255 (Figure 6.9b).

The final dataset, consisting of 356 spectrograms and their corresponding binary masks, was divided into an 80-20 train-test split. These data splits were processed using a custom `ImageSegmentationGenerator` to create batches of spectrograms and their associated masks for model training. The U-Net architecture, a well-established framework for image segmentation tasks (Section 6.3.2), was employed as the segmentation model. The model was compiled using the Adam optimiser with a learning rate of 10^{-5}, and binary cross-entropy was utilised as the loss function. Model performance was evaluated using binary accuracy and the binary intersection over union (IoU) metric.

6.6.2 Results

The model was trained for 500 epochs, and the results were evaluated on the test set. The performance metrics yielded an overall loss of 38.27%, binary accuracy of 93.97%, and binary IoU of 0.49. Figure 6.10 provides a visual comparison of the original spectrograms, the ground-truth masks, and the masks predicted by the U-Net model.



(a) The *Image Labeler* program is used to manually annotate the region of interest in the spectrogram, with the mask highlighted in light blue.

(b) The binary mask output corresponding to the spectrogram in subfigure (a). The mask is a binary image, where the background is represented by 0, and the region of interest is represented by 255.

Figure 6.9: The image masking process using MATLAB's *Image Labeler*.

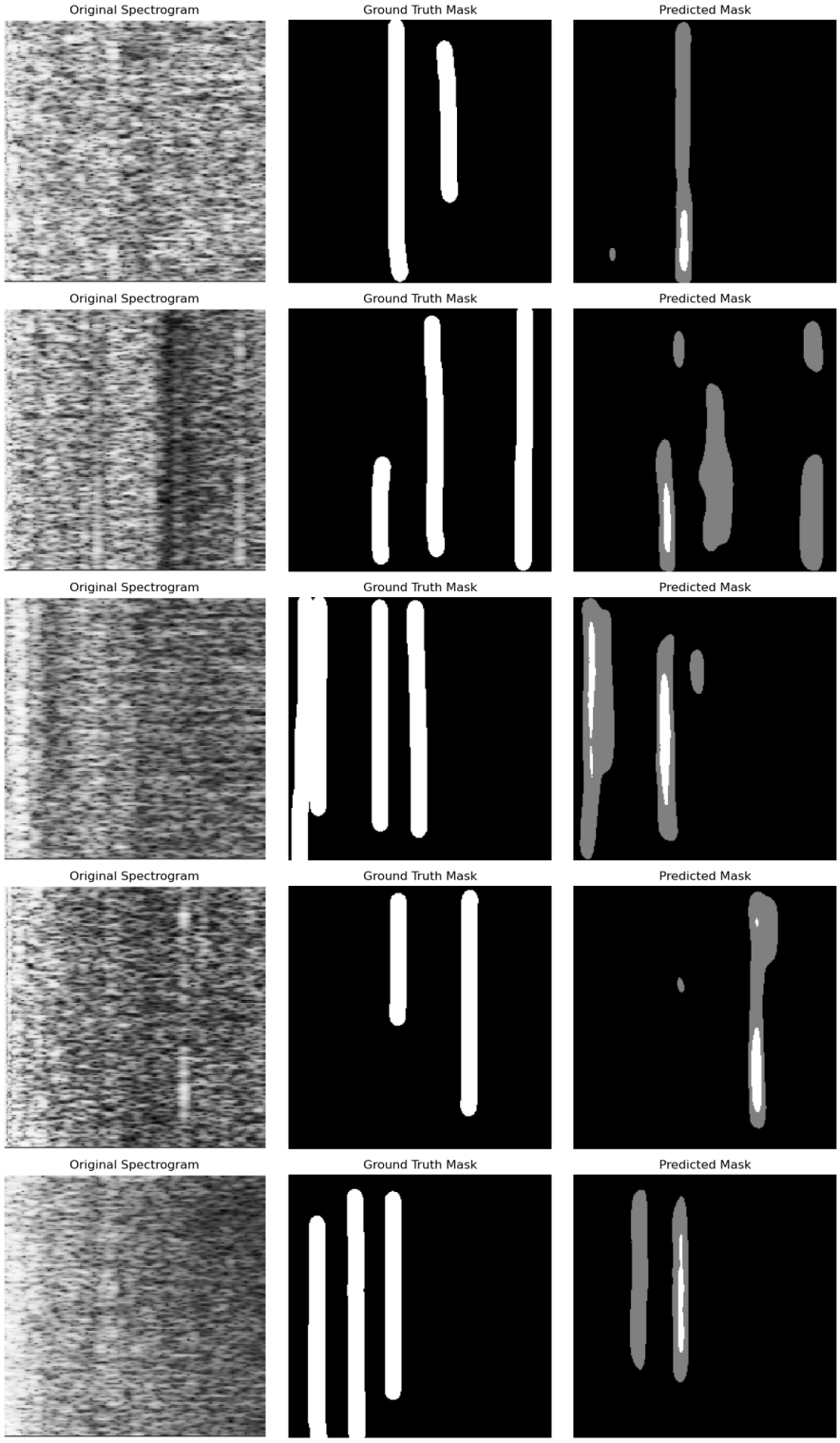


Figure 6.10: Comparison of original spectrograms, ground truth masks, and predicted masks after the U-Net model was trained for 500 epochs.

6.6.3 Discussion

The results of the U-Net model, with a binary accuracy of 93.97% and a relatively low binary IoU score of 0.49, suggest that the predicted masks were effective at identifying many narrowband events but fell short of capturing all instances. As illustrated in Figure 6.10, the model often identified regions of interest with reasonable accuracy, but some events were missed entirely, and random artifacts occasionally appeared in areas where they were not expected.

One major challenge in this experiment was the resolution of the spectrograms. The FFT length of 1024, as specified in Table 3.2, may have provided insufficient frequency resolution to clearly represent narrowband events. The low resolution made it difficult for the model to distinguish these events from the surrounding noise, resulting in less precise masks. This reinforces the importance of high-quality input data for effective model training – in other words: "garbage in, garbage out."

Another contributing factor was the quality of the ground-truth labels. Manual annotation was conducted by myself, a student of software engineering, rather than a domain expert in sonar processing. While this may have been sufficient for preliminary experimentation, annotations made by a skilled sonar technician would be more precise, especially with higher-resolution spectrograms. Improved ground-truth labels would provide the model with better guidance during training, potentially leading to more accurate segmentation outcomes.

Additionally, the dataset size – 356 annotated spectrograms – was a major limiting factor in this experiment. This data volume is far below the scale of datasets typically used for state-of-the-art image segmentation models, which often consist of tens of thousands of images. While the limited dataset was sufficient for initial experimentation, future work should aim to increase the number of annotated samples to at least a few thousand to improve model generalisation and robustness.

For future work, several promising avenues could be pursued. First, improvements in both the input data (higher-resolution spectrograms) and the labelling process (expert-provided masks) could address many of the challenges observed in this experiment. Secondly, the use of automatic masking techniques presents an exciting opportunity to overcome the time, cost, and expertise constraints associated with manual labelling. Two promising methods in this area are the HIDE & SEEK algorithm and track detection algorithms.

The HIDE & SEEK algorithm, developed by Akeret et al. in 2017 [204], consists of two related Python packages designed for simulating and processing radio survey data. The SEEK package, in particular, removes artifacts from radio-frequency interference (RFI) and generates masks identifying these interference areas. Although RFI and noise in underwater acoustic data differ in nature, they share common characteristics, and SEEK has shown promising results when applied to audio data. Initial results from applying SEEK to DeepShip spectrograms (Figure 6.11) are encouraging, though further refinement and testing are required to optimise this approach – work that could not be fully explored within

the time constraints of this project.

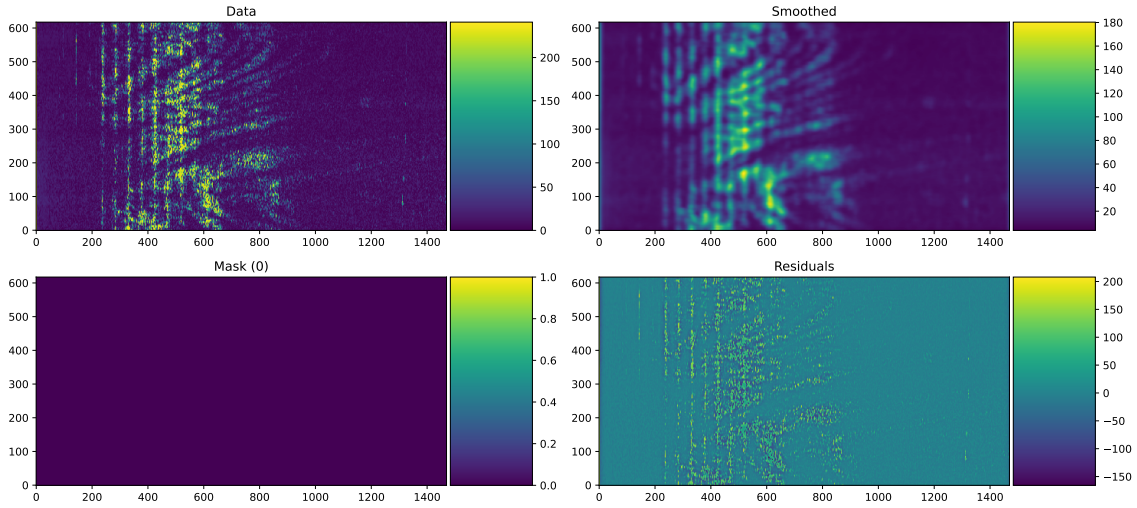


Figure 6.11: Preliminary results from applying the SEEK algorithm to spectrograms. The output shows the potential of SEEK for identifying regions of interest in spectrograms, although further fine-tuning is required.

The second promising direction lies in exploring track detection algorithms. These algorithms are specifically designed to detect narrowband tracks in spectrograms, which makes them highly relevant to this task. A brief review of track detection algorithms was presented earlier in Table 2.1. These algorithms could provide a means of approximating ground-truth masks without manual intervention. Leveraging these algorithms could significantly accelerate the data preparation process while maintaining a high standard of accuracy.

Finally, a key limitation of this study was the absence of an evaluation step that applied the generated masks to the baseline CNN-LSTM model to assess their impact on classification performance. Conducting such an evaluation would offer valuable insights into the practical utility of masking-based denoising techniques for underwater acoustic target recognition tasks. Future research should prioritise this step to fully explore the potential of these approaches.

6.6.4 Conclusion

The U-Net model demonstrated strong potential for identifying key narrowband events in spectrograms, achieving a binary accuracy of 93.97%. However, the relatively low binary IoU score of 0.49 underscores the need for improvements in mask precision and the detection of all narrowband events. These findings suggest that the resolution of the spectrograms and the quality of the manual labelling process played critical roles in limiting the model's performance. Future efforts should prioritise the use of higher-resolution spectrograms and expert-provided ground-truth annotations to ensure that the regions of interest are accurately captured. Expanding the dataset size is also recommended to enhance model

generalisation and robustness.

Furthermore, the exploration of automatic masking techniques offers a promising path forward. Algorithms like HIDE & SEEK and track detection methods could significantly reduce the time and effort required for manual annotation while maintaining accuracy. Preliminary results using SEEK were encouraging, though additional refinement and testing are needed to optimise its application to underwater acoustic data. By combining improved input data with advanced automatic masking techniques, future research could produce more accurate segmentation masks in the aim of enhancing classification performance in underwater acoustic target recognition tasks.

6.7 Concluding remarks

The challenges of denoising underwater acoustic signals underscore the complexities of adapting machine learning techniques originally designed for image data to a vastly different domain. In the realm of images, noise predominantly arises from the sensor itself—random fluctuations inherent to the image capture process. In contrast, noise in underwater signals stems from a multitude of external sources, including marine wildlife, ship traffic, industrial activity, and environmental factors like waves and water currents. This fundamental difference is likely a key reason why our first experiment, applying the Noise2Noise framework to spectrograms, failed to produce meaningful results. Techniques developed for reducing sensor noise are ill-suited to address the diverse and dynamic noise conditions present in underwater environments.

Noise2Noise, while highly effective for image denoising, relies on specific assumptions—namely, that paired inputs share the same underlying signal but have uncorrelated noise. These assumptions are difficult to replicate in underwater acoustics, where the noise is influenced by countless variables that can change over time and space. The experiment highlighted the limitations of current approaches in adapting image-based methodologies to the underwater domain, particularly when paired data is unavailable or when noise characteristics deviate significantly from the assumptions underpinning these frameworks.

On the other hand, the masking-based approach presents a promising alternative for underwater signal denoising. By focusing on isolating regions of interest in spectrograms, the masking method bypasses some of the challenges associated with directly modeling noise. While the results of our masking experiments were not definitive, they indicate potential for future development. Continued exploration into automatic masking techniques, such as integrating advanced track detection algorithms or refining U-Net-based segmentation methods, could lead to significant advancements in underwater acoustic signal processing.

Ultimately, our findings reaffirm the dominance of traditional signal processing methods in underwater acoustics. Machine learning approaches, while transformative in other fields, must be tailored more carefully to the unique characteristics of this domain. The masking method offers a glimmer of hope for achieving this adaptation, and future work should focus on combining the strengths of signal processing with the adaptability of machine learning.

By bridging these two paradigms, the potential for enhancing underwater acoustic signal classification and denoising could finally be realised.

6.7.1 Future work

In addition to the core work presented here, several other avenues of exploration were considered for this chapter but ultimately not pursued due to time constraints. One of the most promising of these avenues was the use of the Noise2Void (N2V) algorithm for denoising underwater acoustic spectrograms.

The Noise2Noise framework, as demonstrated in Section 6.5.1, is highly effective at performing denoising without the need for clean target images. However, it has a major limitation: it requires paired noisy images with independent noise distributions for the algorithm to work. This dependency on paired data makes the approach less applicable in scenarios where only noisy data is available. Seeing as this is the case for most real-world applications – including our own with underwater acoustic data – the Noise2Noise framework may be more theoretically impressive than practical.

The Noise2Void algorithm, a successor to Noise2Noise, addresses this limitation by employing a truly self-supervised approach [205]. Unlike Noise2Noise, which relies on the properties of paired noisy images, Noise2Void exploits the inherent structure within the image itself to perform denoising. Specifically, the algorithm operates on the assumption that *signal has structure and noise does not.* In practice, this means that the algorithm can predict the true signal for a given pixel by examining its surrounding context, while it is unable to predict the noise, which lacks a structured pattern. To achieve this, Noise2Void uses a technique known as *blind spot learning*, where one pixel is excluded from the input image (the "blind spot") and the algorithm attempts to predict its value based on the surrounding pixels.

This method has demonstrated significant promise in the literature and has been successfully applied to various problems, though mostly related to medical imaging [206]–[210]. Many improved versions of N2V have been published, and work on blind spot networks is still an open research area [211]–[214]. Given its potential, we aimed to investigate its application to the domain of underwater acoustic signal processing, specifically for denoising spectrograms.

To explore this, we implemented the Noise2Void algorithm in Keras and trained it for 100 epochs, using the BSD training set for training and the BSD test set for validation. The denoising results on the BSD dataset, shown in Figure 6.12, qualitatively indicate that our implementation of Noise2Void produced quite accurate denoising results, which look visually comparable to those of both supervised methods and the Noise2Noise framework.

However, when we applied the algorithm to the DeepShip spectrograms, the results were suboptimal. As shown in Figure 6.13, the denoised spectrograms exhibited poor quality, and overall simply achieved a mild blurring effect. Upon further investigation, we determined that these issues were primarily due to issues in our implementation and mismatches in the input size, which were not adequately addressed within the time frame

of this thesis.

With further refinement of the algorithm implementation, the Noise2Void algorithm could be successfully applied to the task of denoising underwater acoustic spectrograms, and could be a real, practically-applicable model for underwater acoustic signal denoising given that it only relies on single noisy image inputs.

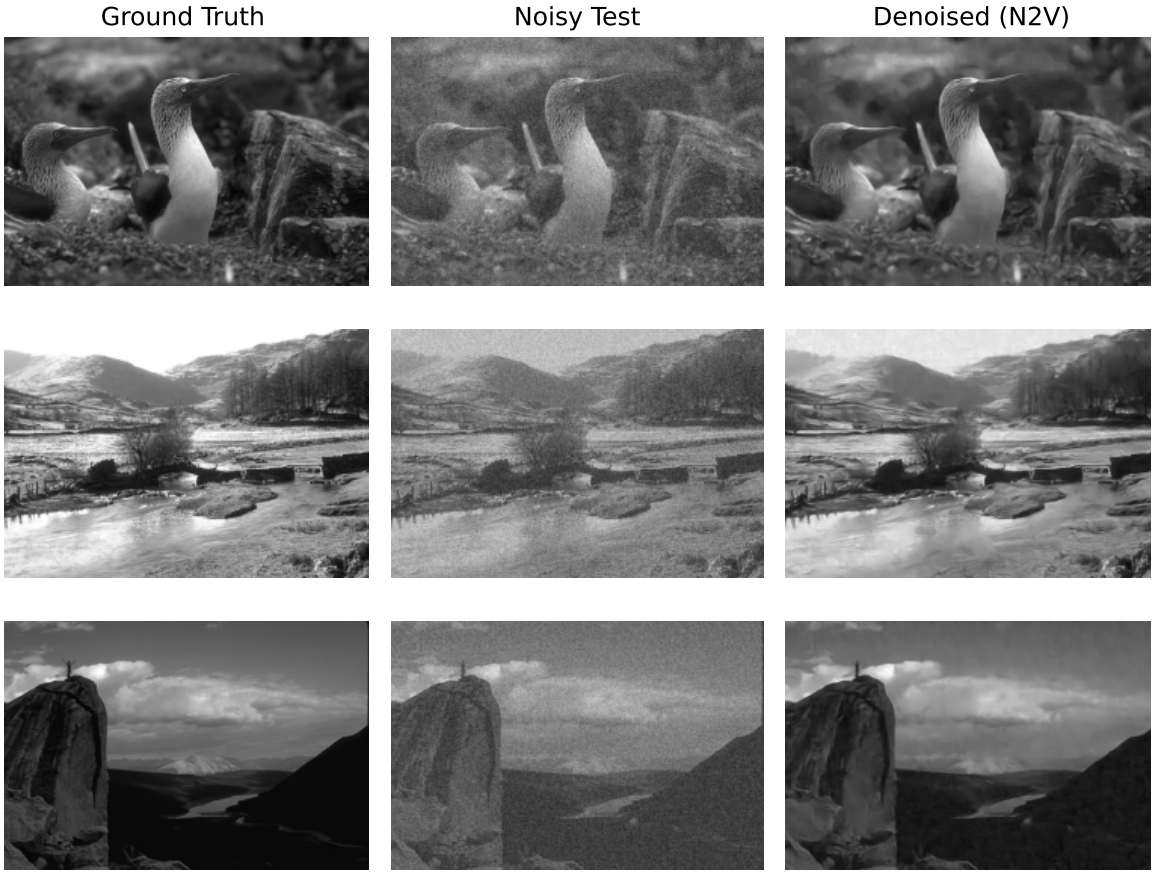


Figure 6.12: Denoising results on the BSD dataset using the Noise2Void algorithm.

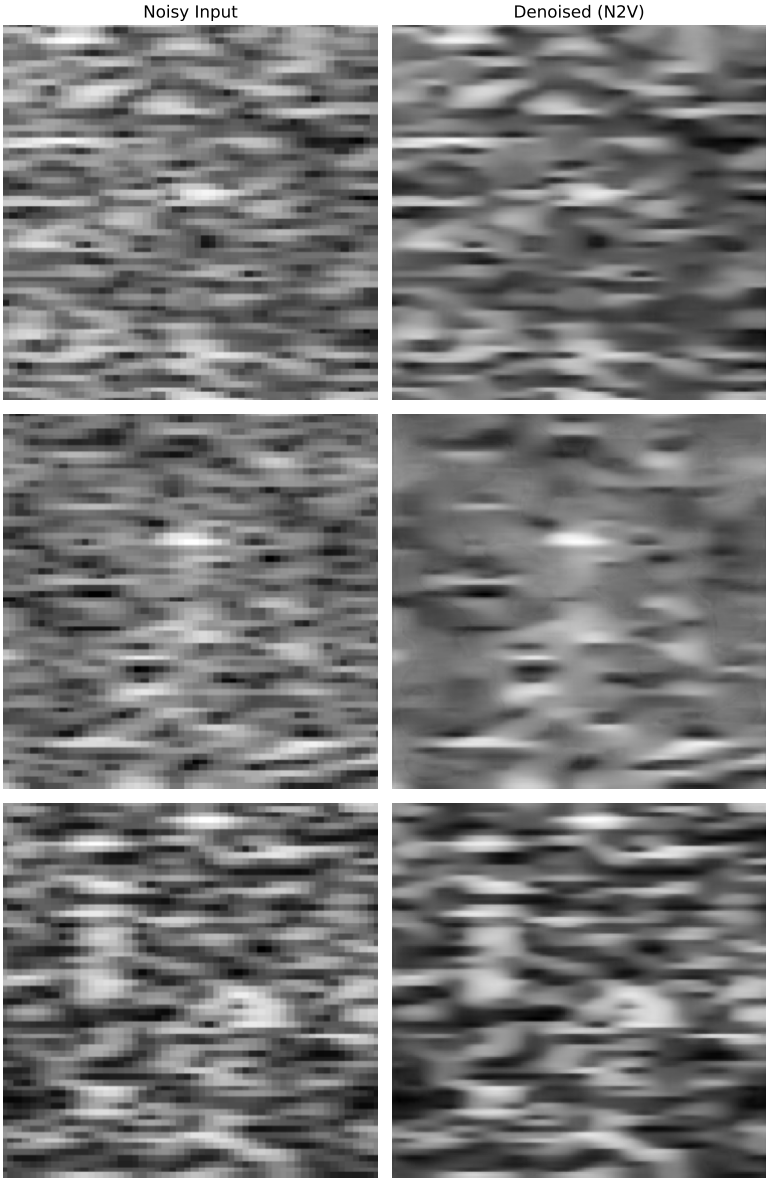


Figure 6.13: Denoising results on the DeepShip spectrograms using the Noise2Void algorithm. The poor results are attributed to challenges with the input size and algorithm implementation, which were not resolved within the scope of this thesis.

Chapter 7

Conclusion

This thesis explored the role of preprocessing techniques in enhancing feature representations for UATR, with a focus on three key methods: normalisation, detrending, and denoising. Using the DeepShip dataset and a CNN-LSTM architecture as the benchmark classifier, we systematically evaluated how these preprocessing approaches could refine spectrogram inputs to improve classification performance.

The hybrid CNN-LSTM model was selected as the baseline classifier due to its ability to capture both spatial and temporal features from spectrograms effectively. The DeepShip dataset was segmented into three-second intervals and organised into 10 folds for k-fold cross-validation. The model achieved a baseline accuracy of 63.41%, establishing a robust reference point for assessing the influence of the preprocessing techniques on classification performance.

The normalisation experiment demonstrated that both channel-based and global normalisation had minimal impact on classification performance. This is likely due to the consistent feature scales already present in the DeepShip dataset, resulting from its controlled recording conditions and the logarithmic transformation inherent in power spectrogram preprocessing. These findings suggest that normalisation might have a greater impact on datasets with more variability, such as those collected using dynamic towed arrays or in acoustically diverse environments.

The detrending experiment assessed the effectiveness of the ℓ_1 detrending algorithm in suppressing long-term trends and isolating short-term fluctuations within spectrograms. Unfortunately, detrending consistently led to a decline in classification accuracy across all tested parameter values. Lower α values introduced excessive smoothing, erasing transient features essential for vessel classification, while higher α values inadequately suppressed broadband noise. Additionally, the erratic accuracy-loss curves observed during training indicated that detrending disrupted the spatial and temporal integrity of the spectrograms, impairing the CNN-LSTM model's ability to converge effectively. These results suggest that while ℓ_1 detrending has proven effective in other domains, it is not well-suited for UATR tasks in its current form. Future research should investigate alternative detrending approaches, such as wavelet-based methods, and evaluate their compatibility with diverse deep learning architectures to better understand their potential in this domain.

The denoising experiments examined both the Noise2Noise framework and masking-based approaches on the DeepShip dataset. Noise2Noise performed well on natural images but failed to produce meaningful results on underwater acoustic spectrograms due to the framework's reliance on paired noisy inputs with uncorrelated noise – an assumption that is difficult to approximate in the underwater acoustic domain, where noise conditions vary widely. In contrast, the masking-based approach showed some promise, achieving a high binary accuracy of 93.97% but a lower intersection-over-union score of 0.49. This indicates that while the U-Net segmentation model accurately identified many regions of interest, it failed to capture all relevant segments consistently. Future research could improve these results by using higher-resolution spectrograms, leveraging expert-curated ground-truth masks, or incorporating advanced automated masking techniques.

This thesis highlights the challenges of adapting preprocessing techniques to the underwater acoustic domain and underscores the importance of tailoring methods to the unique characteristics of sonar data. While normalisation, detrending, and denoising each showed limitations, the insights gained provide a valuable foundation for refining preprocessing strategies in future UATR research.

This thesis also offers a broader reflection on the complexities of applying machine learning to underwater acoustics, particularly when treating spectrograms as analogous to natural images. Unlike natural images, which maintain a relatively consistent structure and interpretation regardless of the capturing device or environmental conditions, spectrograms are inherently variable. The very process of generating a spectrogram introduces numerous degrees of freedom, from the choice of Fourier transform parameters to the preprocessing methods employed. Moreover, natural images typically contain a finite number of interpretable objects – such as a bird, a tree, or a car – that can be verified against ground truth. In contrast, spectrograms represent a continuous time-frequency domain, often encompassing an unbounded number of overlapping, unidentifiable acoustic features, making the extraction of meaningful patterns significantly more challenging.

These fundamental differences underscore why applying deep learning to underwater acoustic spectrograms is more complex and inconsistent than to natural image data. Preprocessing techniques like normalisation, which are well-established and universally effective in image processing, exhibit unpredictable outcomes when applied to spectrograms. For example, the normalisation experiments in this thesis demonstrated that the recording conditions of the DeepShip dataset – using a fixed hydrophone recording in a 1km radius – reduced the efficacy of standard normalisation techniques. Such an issue, absent in natural image datasets, highlights just one of the unique and unpredictable challenges of preprocessing for UATR.

At the outset of this thesis, the slower progress in UATR compared to image-based machine learning posed a significant question: why has deep learning not yet overtaken traditional signal processing as the preferred method for underwater acoustic target recognition? This work provides direction for an answer. The variability and uncertainty inherent in underwater acoustic data demand the human qualities of inference and problem-solving

– abilities that machines cannot yet replicate. The complexity and unique challenges of underwater acoustics render current image-based methodologies inadequate for this domain. Bridging this gap will require years of dedicated research and the development of preprocessing and machine learning techniques explicitly tailored to the underwater acoustic environment. This thesis serves as an initial step in that direction, offering valuable insights into the limitations and potential of existing approaches.

Appendix A

Case Studies

I have chosen two electives to fulfill my 12 credit points of elective coursework: *ELEC5305: Acoustics, Speech and Signal Processing*, and *ELEC5307: Advanced Signal Processing with Deep Learning*. Below I have outlined how I have met the learning outcomes from each unit of study.

A.1 ELEC5305: Acoustics, Speech and Signal Processing

- LO1: Mastery of analytical and mathematical skills related to acoustic signal processing
  - My thesis began with an in-depth exploration of foundational acoustic signal processing techniques such as the short-time Fourier transform, Mel spectrum, LOFAR spectra, and the Constant Q transform. These methods form the backbone of state-of-the-art approaches in underwater acoustic target recognition. By thoroughly understanding these techniques, I was able to analyse their suitability as feature inputs and build upon them to improve underwater target classification systems.
- LO2: Proficiency in developing signal processing software
  - Using MATLAB's Audio Processing Toolkit I developed algorithms to tackle a variety of acoustic signal processing challenges. For example, I implemented the ℓ_1 detrending algorithm for spectral data in Chapter 5 to better understand the impact of preprocessing methods on classification accuracy.
- LO3: Planning, designing, and reviewing signal processing systems
  - As part of my work, I designed an end-to-end signal processing pipeline tailored for UATR tasks. This pipeline includes the selection of appropriate signal representations, feature extraction techniques, and integration with machine learning models for classification. Each stage of the pipeline is reviewed and evaluated based on its impact on overall system performance.

- LO4: Developing innovative ideas in signal processing systems
  - My thesis incorporates cutting-edge techniques such as self-supervised and unsupervised learning to complement traditional signal processing methods. For instance, I applied the Noise2Noise framework to underwater acoustic spectrograms and explored masking-based denoising techniques in Chapter 6.
- LO5: Communicating signal processing practice effectively
  - I communicated the methodologies and findings of my work in a clear and accessible manner through informative visualisations and comprehensive documentation. These materials were prepared for both academic audiences and industry stakeholders, ensuring the applicability and relevance of my work.
- LO6: Contributing to team-based projects
  - As an intern with the Underwater Systems team at Thales, I worked collaboratively with colleagues, presenting weekly updates and incorporating their feedback to refine the project's direction. This experience not only strengthened my technical skills but also underscored the importance of teamwork in advancing research and development goals.

A.2 ELEC5307: Advanced Signal Processing with Deep Learning

- LO1: Using appropriate software platforms for multi-dimensional signal processing tasks
  - My thesis integrates Python libraries such as TensorFlow, Keras, and Scikit-learn for implementing and training deep learning models. These are complemented by MATLAB, which I used for preprocessing tasks such as spectrogram generation and signal transformation. This combination allowed me to bridge traditional signal processing techniques with advanced deep learning methods effectively.
- LO2: Applying machine learning and deep learning methods to multi-dimensional signal processing
  - Throughout my research, I designed and implemented a variety of deep learning architectures to address multi-dimensional signal processing challenges in underwater acoustic target recognition. These include convolutional autoencoders for feature extraction, CNNs and U-Nets for image-based tasks such as denoising and segmentation, and hybrid models like CNN-LSTM to capture both spatial and temporal dependencies in spectrogram data.
- LO3: Using existing machine learning and deep learning toolboxes

  - My project extensively leveraged pre-built libraries and toolboxes to implement complex models and streamline experimentation. In Python, TensorFlow and Keras facilitated the construction and training of deep learning architectures, while MATLAB was used for signal preprocessing and transformation. This allowed me to focus on adapting and optimising these tools for specific UATR tasks, ensuring their applicability in real-world scenarios.

- LO4: Reporting results professionally

  - The outcomes of my research were communicated through structured presentations and this thesis. These documents adhere to academic and professional standards, incorporating clear visualisations, well-structured discussions, and actionable conclusions to ensure clarity and impact for a diverse audience of academic researchers and industry stakeholders.

References

[1] H. Niu, X. Li, Y. Zhang, and X. Ji, "Advances and applications of machine learning in underwater acoustics," *Intelligent Marine Technology and Systems*, vol. 1, no. 1, Oct. 20, 2023. DOI: 10.1007/s44295-023-00005-0.

[2] A. D. Waite, *Sonar for practising engineers*, 3rd ed. Chichester: Wiley, 2002, 298 pp., ISBN: 978-0-471-49750-9.

[3] L. C. F. Domingos, P. E. Santos, P. E. Santos, *et al.*, "A survey of underwater acoustic data classification methods using deep learning for shoreline surveillance," *Sensors*, vol. 22, no. 6, pp. 2181–2181, Mar. 11, 2022. DOI: 10.3390/s22062181.

[4] M. A. Aslam, L. Zhang, X. Liu, *et al.*, "Underwater sound classification using learning based methods: A review," *Expert Systems with Applications*, vol. 255, p. 124 498, Dec. 2024. DOI: 10.1016/j.eswa.2024.124498.

[5] D. Neupane and J. Seok, "A review on deep learning-based approaches for automatic sonar target recognition," *Electronics*, vol. 9, no. 11, p. 1972, Nov. 22, 2020. DOI: 10.3390/electronics9111972.

[6] X. Luo, L. Chen, H. Zhou, and H. Cao, "A survey of underwater acoustic target recognition methods based on machine learning," *Journal of Marine Science and Engineering*, vol. 11, no. 2, p. 384, Feb. 9, 2023. DOI: 10.3390/jmse11020384.

[7] R. J. Urick, *Principles of underwater sound*, 2. ed. New York: McGraw-Hill, 1975, 384 pp., ISBN: 978-0-07-066086-1.

[8] M. Mersenne and R. E. Chapman, *Harmonie Universelle: The Books on Instruments*, Softcover reprint of the original 1st ed. 1957 edition. Springer, Jan. 1, 1957, 608 pp., ISBN: 978-94-017-5781-2.

[9] S. I. Newton, *The Principia: Mathematical Principles of Natural Philosophy*. Createspace Independent Publishing Platform, Jul. 5, 2013, 464 pp., ISBN: 978-1-4905-9215-2.

[10] O. Darrigol, "The acoustic origins of harmonic analysis," *Archive for History of Exact Sciences*, vol. 61, no. 4, pp. 343–424, Jul. 2007. DOI: 10.1007/s00407-007-0003-9.

[11] W. Dixon Ward, "Musical perception," in *Foundations of Modern Auditory Theory*, vol. 1, Academic Press, 1970, p. 438.

[12] H. Helmholtz, *On the Sensations of Tone*, 2nd ed. edition. New York: Dover Publications, Jun. 1, 1954, 608 pp., ISBN: 978-0-486-60753-5.

[13] J. W. S. B. Rayleigh, *The Theory of Sound*, 1st edition. Cambridge: Cambridge University Press, Jun. 2, 2011, 344 pp., ISBN: 978-1-108-03220-9.

[14] R. Vaccaro, "The past, present, and the future of underwater acoustic signal processing," *IEEE Signal Processing Magazine*, vol. 15, no. 4, pp. 21–51, Jul. 1998. DOI: 10.1109/79.689583.

[15] J. Dinneen. "Reginald fessenden and the invention of sonar," Science History Institute. (May 19, 2020).

[16] O. C. Rodríguez, *Fundamentals of Underwater Acoustics*. Cham: Springer Nature Switzerland, 2023. DOI: 10.1007/978-3-031-31319-6.

[17] G. Helgason. "ASDIC / sonar." (Oct. 9, 2011).

[18] National Oceanic and Atmospheric Association. "What is SOFAR?" (Jun. 16, 2024).

[19] H. Morin. "History of the SOFAR channel," Discovery of Sound in the Sea. ().

[20] C. Erbe, "Underwater noise of whale-watching boats and potential effects on killer whales based on an acoustic impact model," *Marine Mammal Science*, vol. 18, no. 2, pp. 394–418, Apr. 2002. DOI: 10.1111/j.1748-7692.2002.tb01045.x.

[21] H. Slabbekoorn, N. Bouton, I. Van Opzeeland, A. Coers, C. Ten Cate, and A. N. Popper, "A noisy spring: The impact of globally rising underwater sound levels on fish," *Trends in Ecology & Evolution*, vol. 25, no. 7, pp. 419–427, Jul. 2010. DOI: 10.1016/j.tree.2010.04.005.

[22] L. E. Wysocki, S. Amoser, and F. Ladich, "Diversity in ambient noise in european freshwater habitats: Noise levels, spectral profiles, and impact on fishes," *The Journal of the Acoustical Society of America*, vol. 121, no. 5, pp. 2559–2566, May 1, 2007. DOI: 10.1121/1.2713661.

[23] SINAY. "Diving into the depths: Understanding underwater acoustics and its impact on marine life.," https://sinay.ai/. (Mar. 27, 2024).

[24] C. C. Leroy, S. P. Robinson, and M. J. Goldsmith, "A new equation for the accurate calculation of sound speed in all oceans," *The Journal of the Acoustical Society of America*, vol. 124, no. 5, pp. 2774–2782, Nov. 1, 2008. DOI: 10.1121/1.2988296.

[25] D. A. Abraham, *Underwater Acoustic Signal Processing: Modeling, Detection, and Estimation* (Modern Acoustics and Signal Processing Ser). Cham: Springer International Publishing AG, 2019, 1 p., ISBN: 978-3-319-92981-1.

[26] D. R. Jackson and M. D. Richardson, *High-Frequency Seafloor Acoustics*. New York, NY: Springer New York, 2007. DOI: 10.1007/978-0-387-36945-7.

[27] L. Bjørnø, T. H. Neighbors, and D. Bradley, *Applied underwater acoustics*. Amsterdam, Netherlands: Elsevier, 2017, 964 pp., ISBN: 978-0-12-811240-3.

[28] D. Ross, *Mechanics of underwater noise*. New York: Pergamon Press, 1976.

[29] A. Zak, "Ships classification basing on acoustic signatures," *WSEAS Transactions on Signal Processing*, vol. 4, no. 4, pp. 137–149, Apr. 2008.

[30] C. Chin-Hsing, L. Jiann-Der, and L. Ming-Chi, "Classification of underwater signals using wavelet transforms and neural networks," *Mathematical and Computer Modelling*, vol. 27, no. 2, pp. 47–60, Jan. 1998. DOI: 10.1016/S0895-7177(97)00259-8.

[31] S. J. Malinowski, I. Gloza, and J. Domagalski, "Underwater noise radiated by ships, their propulsion and auxiliary machinery, and propellers," *Hydroacoustics*, vol. 4, pp. 165–168, 2001.

[32] C. Knowlton. "What are common underwater sounds?" Discovery of Sound in the Sea. ().

[33] G. M. Wenz, "Review of underwater acoustics research: Noise," *The Journal of the Acoustical Society of America*, vol. 51, no. 3, pp. 1010–1024, Mar. 1, 1972. DOI: 10.1121/1.1912921.

[34] R. K. Andrew, B. M. Howe, J. A. Mercer, and M. A. Dzieciuch, "Ocean ambient sound: Comparing the 1960s with the 1990s for a receiver off the california coast," *Acoustics Research Letters Online*, vol. 3, no. 2, pp. 65–70, Apr. 2002. DOI: 10.1121/1.1461915.

[35] H. Medwin and M. M. Beaky, "Bubble sources of the knudsen sea noise spectra," *The Journal of the Acoustical Society of America*, vol. 86, no. 3, pp. 1124–1130, Sep. 1, 1989. DOI: 10.1121/1.398104.

[36] G. J. Franz, "Splashes as sources of sound in liquids," *The Journal of the Acoustical Society of America*, vol. 31, no. 8, pp. 1080–1096, Aug. 1, 1959. DOI: 10.1121/1.1907831.

[37] P. H. Dahl, J. H. Miller, D. H. Cato, and R. K. Andrew, "Underwater ambient noise," *Acoustics Today*, vol. 3, no. 1, p. 23, 2007. DOI: 10.1121/1.2961145.

[38] D. H. Cato and M. J. Bell, *Ultrasonic ambient noise in Australian shallow waters at frequencies up to 200 kHz* (MRL technical report MRL-TR-91-23). 1992, 27 pp., ISBN: 0-646-10798-4.

[39] W. J. Richardson, C. R. Greene, C. I. Malme, and D. H. Thomson, "Marine mammal sounds," in *Marine Mammals and Noise*, Elsevier, 1995, pp. 159–204, ISBN: 978-0-08-057303-8. DOI: 10.1016/B978-0-08-057303-8.50010-0.

[40] S. Patek and S. J. Longo, "Evolutionary biomechanics: The pathway to power in snapping shrimp," *Current Biology*, vol. 28, no. 3, R115–R117, Feb. 2018. DOI: 10.1016/j.cub.2017.12.033.

[41] D. Lohse, B. Schmitz, and M. Versluis, "Snapping shrimp make flashing bubbles," *Nature*, vol. 413, no. 6855, pp. 477–478, Oct. 2001. DOI: 10.1038/35097152.

[42] R. Ritzmann, "Snapping behavior of the shrimp *Alpheus californiensis*," *Science*, vol. 181, no. 4098, pp. 459–460, Aug. 3, 1973. DOI: 10.1126/science.181.4098.459.

[43] M. J. Bianco, P. Gerstoft, J. Traer, *et al.*, "Machine learning in acoustics: Theory and applications," *The Journal of the Acoustical Society of America*, vol. 146, no. 5, pp. 3590–3628, Nov. 1, 2019. DOI: 10.1121/1.5133944.

[44] X. Su, I. Ullah, X. Liu, and D. Choi, "A review of underwater localization techniques, algorithms, and challenges," *Journal of Sensors*, vol. 2020, pp. 1–24, Jan. 13, 2020. DOI: 10.1155/2020/6403161.

[45] Y. Wang and H. Peng, "Underwater acoustic source localization using generalized regression neural network," *The Journal of the Acoustical Society of America*, vol. 143, no. 4, pp. 2321–2331, Apr. 1, 2018. DOI: 10.1121/1.5032311.

[46] B. Zhang, Y. Hu, H. Wang, and Z. Zhuang, "Underwater source localization using TDOA and FDOA measurements with unknown propagation speed and sensor parameter errors," *IEEE Access*, vol. 6, pp. 36 645–36 661, 2018. DOI: 10.1109/ACCESS.2018.2852636.

[47] E. Ozanich, P. Gerstoft, and H. Niu, "A feedforward neural network for direction-of-arrival estimation," *The Journal of the Acoustical Society of America*, vol. 147, no. 3, pp. 2035–2048, Mar. 1, 2020. DOI: 10.1121/10.0000944.

[48] Y. Liu, H. Chen, and B. Wang, "DOA estimation based on CNN for underwater acoustic array," *Applied Acoustics*, vol. 172, p. 107 594, Jan. 2021. DOI: 10.1016/j.apacoust.2020.107594.

[49] X. Li, J. Chen, J. Bai, *et al.*, "Deep learning-based DOA estimation using CRNN for underwater acoustic arrays," *Frontiers in Marine Science*, vol. 9, p. 1 027 830, Nov. 10, 2022. DOI: 10.3389/fmars.2022.1027830.

[50] X. Cao, X. Zhang, Y. Yu, and L. Niu, "Deep learning-based recognition of underwater target," in *2016 IEEE International Conference on Digital Signal Processing (DSP)*, Beijing, China: IEEE, Oct. 2016, pp. 89–93, ISBN: 978-1-5090-4165-7. DOI: 10.1109/ICDSP.2016.7868522.

[51] P. Cauchy, K. J. Heywood, N. D. Merchant, B. Y. Queste, and P. Testor, "Wind speed measured from underwater gliders using passive acoustics," *Journal of Atmospheric and Oceanic Technology*, vol. 35, no. 12, pp. 2305–2321, Dec. 2018. DOI: 10.1175/JTECH-D-17-0209.1.

[52] B. Mishachandar and S. Vairamuthu, "Diverse ocean noise classification using deep learning," *Applied Acoustics*, vol. 181, p. 108 141, Oct. 2021. DOI: 10.1016/j.apacoust.2021.108141.

[53] A. Amick. "How submarine sonarmen tirelessly hunt for enemies they can't even see," The Warzone. (Nov. 27, 2020).

[54] Carnegie Mellon University. "Human and machine minds," Human and Machine Minds. (Oct. 2001).

[55] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach* (Pearson Series in Artificial Intelligence), Fourth Edition, in collab. with M.-w. Chang, J. Devlin, A. Dragan, *et al.* Hoboken, NJ: Pearson, 2021, 1115 pp., ISBN: 978-0-13-461099-3.

[56] S. Kamal, S. K. Mohammed, P. R. S. Pillai, and M. H. Supriya, "Deep learning architectures for underwater target recognition," in *2013 Ocean Electronics (SYMPOL)*, Kochi, India: IEEE, Oct. 2013, pp. 48–54, ISBN: 978-93-80095-45-5. DOI: 10.1109/SYMPOL.2013.6701911.

[57] J. Abel, H. Lee, and A. Lowell, "An image processing approach to frequency tracking (application to sonar data)," in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, CA, USA: IEEE, 1992, 561–564 vol.2, ISBN: 978-0-7803-0532-8. DOI: 10.1109/ICASSP.1992.225995.

[58] T. A. Lampert and S. E. O'Keefe, "A survey of spectrogram track detection algorithms," *Applied Acoustics*, vol. 71, no. 2, pp. 87–100, Feb. 2010. DOI: 10.1016/j.apacoust.2009.08.007.

[59] H. Zhang, C. Li, H. Wang, J. Wang, and F. Yang, "Frequency line extraction on low SNR lofargram using principal component analysis," in *2018 14th IEEE International Conference on Signal Processing (ICSP)*, Beijing, China: IEEE, Aug. 2018, pp. 455–459, ISBN: 978-1-5386-4673-1. DOI: 10.1109/ICSP.2018.8652411.

[60] X. Luo and Z. Shen, "A sensing and tracking algorithm for multiple frequency line components in underwater acoustic signals," *Sensors*, vol. 19, no. 22, p. 4866, Nov. 8, 2019. DOI: 10.3390/s19224866.

[61] Y. Han, Y. Li, Q. Liu, and Y. Ma, "DeepLofargram: A deep learning based fluctuating dim frequency line detection and recovery," *The Journal of the Acoustical Society of America*, vol. 148, no. 4, pp. 2182–2194, Oct. 1, 2020. DOI: 10.1121/10.0002172.

[62] P. Li, X. Liu, K. J. Palmer, *et al.*, "Learning deep models from synthetic data for extracting dolphin whistle contours," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–10, ISBN: 978-1-72816-926-2. DOI: 10.1109/IJCNN48605.2020.9206992.

[63] C. Huang, K. Yang, Q. Yang, and H. Zhang, "Line spectrum extraction based on autoassociative neural networks," *JASA Express Letters*, vol. 1, no. 1, p. 016 003, Jan. 1, 2021. DOI: 10.1121/10.0003038.

[64] X. Luo and Z. Shen, "A space-frequency joint detection and tracking method for line-spectrum components of underwater acoustic signals," *Applied Acoustics*, vol. 172, p. 107 609, Jan. 2021. DOI: 10.1016/j.apacoust.2020.107609.

[65] D. Ju, C. Chi, Z. Li, Y. Li, C. Zhang, and H. Huang, "Deep-learning-based line enhancer for passive sonar systems," *IET Radar, Sonar & Navigation*, vol. 16, no. 3, pp. 589–601, Mar. 2022. DOI: 10.1049/rsn2.12205.

[66] X. Li, D. Wang, Y. Tian, and X. Kong, "A method for extracting interference striations in lofargram based on decomposition and clustering," *IET Image Processing*, vol. 17, no. 6, pp. 1951–1958, May 2023. DOI: 10.1049/ipr2.12768.

[67] Z. Li, J. Guo, and X. Wang, "Joint detection and reconstruction of weak spectral lines under non-gaussian impulsive noise with deep learning," *Remote Sensing*, vol. 15, no. 13, p. 3268, Jun. 25, 2023. DOI: 10.3390/rs15133268.

[68] H. Zhou, X. Luo, and L. Chen, "A weak acoustic signal line-spectrum detection method based on stochastic resonance," in *2023 6th International Conference on Information Communication and Signal Processing (ICICSP)*, Xi'an, China: IEEE, Sep. 23, 2023, pp. 471–475, ISBN: 9798350339994. DOI: 10.1109/ICICSP59554.2023.10390708.

[69] Z. Li, J. Guo, and X. Wang, "Weak fluctuating spectral line reconstruction using deep learning," *Journal of Physics: Conference Series*, vol. 2718, no. 1, p. 012 085, Mar. 1, 2024. DOI: 10.1088/1742-6596/2718/1/012085.

[70] P. McCorduck, *Machines who think: a personal inquiry into the history and prospects of artificial intelligence* (An A K Peters book). Boca Raton London New York: CRC Press, 2018, 565 pp., ISBN: 978-1-56881-205-2.

[71] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1, 1950. DOI: 10.1093/mind/LIX.236.433.

[72] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. DOI: 10.1007/BF02478259.

[73] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. DOI: 10.1037/h0042519.

[74] R. Pramoditha. "The concept of artificial neurons (perceptrons) in neural networks," Medium. (Dec. 29, 2021).

[75] M. Minsky and S. A. Papert, *Perceptrons: an introduction to computational geometry*, 2. print. with corr. Cambridge/Mass.: The MIT Press, 1972, 258 pp., ISBN: 978-0-262-13043-1.

[76] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. DOI: 10.1038/323533a0.

[77] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[78] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[79] I. Goodfellow, A. Courville, and Y. Bengio, *Deep learning* (Adaptive computation and machine learning). Cambridge, Massachusetts: The MIT Press, 2016, 1 p., ISBN: 978-0-262-03561-3.

[80] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[81] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL: IEEE, Jun. 2009, pp. 248–255, ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206848.

[82] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft COCO: Common objects in context*, 2014. DOI: 10.48550/ARXIV.1405.0312.

[83] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th Annual International Conference on Machine Learning*, Montreal Quebec Canada: ACM, Jun. 14, 2009, pp. 873–880, ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553486.

[84] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 28, 2015. DOI: 10.1038/nature14539.

[85] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[86] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. DOI: 10.48550/ARXIV.1409.1556.

[87] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. DOI: 10.48550/ARXIV.1512.03385.

[88] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762.

[89] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, 2018. DOI: 10.48550/ARXIV.1810.04805.

[90] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language models are few-shot learners*, 2020. DOI: 10.48550/ARXIV.2005.14165.

[91] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. DOI: 10.48550/ARXIV.2010.11929.

[92] R. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, no. 1, pp. 75–89, Jan. 1988. DOI: 10.1016/0893-6080(88)90023-8.

[93] S. Wang and X. Zeng, "Robust underwater noise targets classification using auditory inspired time–frequency analysis," *Applied Acoustics*, vol. 78, pp. 68–76, Apr. 2014. DOI: 10.1016/j.apacoust.2013.11.003.

[94] Sherin B. M. and Supriya M. H., "Selection and parameter optimization of SVM kernel function for underwater target classification," in *2015 IEEE Underwater Technology (UT)*, Chennai, India: IEEE, Feb. 2015, pp. 1–5, ISBN: 978-1-4799-8300-1. DOI: 10.1109/UT.2015.7108260.

[95] H. Yue, L. Zhang, D. Wang, Y. Wang, and Z. Lu, "The classification of underwater acoustic targets based on deep learning methods," in *Proceedings of the 2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017)*, Sanya, China: Atlantis Press, 2017, ISBN: 978-94-6252-360-9. DOI: 10.2991/caai-17.2017.118.

[96] X. Cao, R. Togneri, X. Zhang, and Y. Yu, "Convolutional neural network with second-order pooling for underwater target classification," *IEEE Sensors Journal*, vol. 19, no. 8, pp. 3058–3066, Apr. 15, 2019. DOI: 10.1109/JSEN.2018.2886368.

[97] F. Liu, T. Shen, Z. Luo, D. Zhao, and S. Guo, "Underwater target recognition using convolutional recurrent neural networks with 3-d mel-spectrogram and data augmentation," *Applied Acoustics*, vol. 178, p. 107 989, Jul. 2021. DOI: 10.1016/j.apacoust.2021.107989.

[98] F. Hong, C. Liu, L. Guo, F. Chen, and H. Feng, "Underwater acoustic target recognition with a residual network and the optimized feature extraction method," *Applied Sciences*, vol. 11, no. 4, p. 1442, Feb. 5, 2021. DOI: 10.3390/app11041442.

[99] F. Hong, C. Liu, L. Guo, F. Chen, and H. Feng, "Underwater acoustic target recognition with ResNet18 on ShipsEar dataset," in *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, Chengdu, China: IEEE, May 7, 2021, pp. 1240–1244, ISBN: 978-1-72817-673-4. DOI: 10.1109/ICET51757.2021.9451099.

[100] V.-S. Doan, T. Huynh-The, and D.-S. Kim, "Underwater acoustic target classification based on dense convolutional neural network," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022. DOI: 10.1109/LGRS.2020.3029584.

[101] S. Feng and X. Zhu, "A transformer-based deep learning network for underwater acoustic target recognition," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022. DOI: 10.1109/LGRS.2022.3201396.

[102] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995. DOI: 10.1007/BF00994018.

[103] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds., red. by J. Kacprzyk, vol. 284, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 65–74, ISBN: 978-3-642-12537-9. DOI: 10.1007/978-3-642-12538-6_6.

[104] D. S. Park, W. Chan, Y. Zhang, *et al.*, "SpecAugment: A simple data augmentation method for automatic speech recognition," 2019. DOI: 10.48550/ARXIV.1904.08779.

[105] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 28, 2006. DOI: 10.1126/science.1127647.

[106] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, May 18, 2015.

[107] V. Badrinarayanan, A. Kendall, and R. Cipolla, *SegNet: A deep convolutional encoder-decoder architecture for image segmentation*, 2015. DOI: 10.48550/ARXIV.1511.00561.

[108] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, 2015. DOI: 10.48550/ARXIV.1502.03167.

[109] A. Ng, *Sparse autoencoder*, Jan. 4, 2010.

[110] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, Jul. 3, 2012. DOI: 10.48550/arXiv.1207.0580.

[111] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[112] Muhaddisa Barat Ali, "Use of dropouts and sparsity for regularisation of autoencoders in deep neural networks," Masters, Bilkent University, Jan. 2015.

[113] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 1, 2013. DOI: 10.1109/tpami.2013.50.

[114] J. Xu, L. Xiang, Q. Liu, *et al.*, "Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 1, pp. 119–130, Jan. 2016. DOI: 10.1109/TMI.2015.2458702.

[115] Y. Chen and J. Shang, "Underwater target recognition method based on convolution autoencoder," in *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, Chongqing, China: IEEE, Dec. 2019, pp. 1–5, ISBN: 978-1-72812-345-5. DOI: 10.1109/ICSIDP47821.2019.9173362.

[116] M. Irfan, Z. Jiangbin, M. Iqbal, and M. H. Arif, "A novel lifelong learning model based on cross domain knowledge extraction and transfer to classify underwater images," *Information Sciences*, vol. 552, pp. 80–101, Apr. 2021. DOI: 10.1016/j.ins.2020.11.048.

[117] M. Irfan, J. Zheng, Jiangbin Zheng, *et al.*, "DeepShip: An underwater acoustic benchmark dataset and a separable convolution based autoencoder for classification," *Expert Systems With Applications*, vol. 183, p. 115 270, Nov. 30, 2021. DOI: 10.1016/j.eswa.2021.115270.

[118] S. Russell, "Inductive learning by machines," *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition*, vol. 64, no. 1, pp. 37–64, 1991.

[119] M. Schmitt, S. A. Ahmadi, Y. Xu, *et al.*, "There are no data like more data: Datasets for deep learning in earth observation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 11, no. 3, pp. 63–97, Sep. 2023. DOI: 10.1109/MGRS.2023.3293459.

[120] David Santos-Domínguez, Soledad Torres-Guijarro, Antonio Cardenal-López, and A. Pena-Gimenez, "ShipsEar: An underwater vessel noise database," *Applied Acoustics*, vol. 113, no. 113, pp. 64–69, Dec. 1, 2016. DOI: 10.1016/j.apacoust.2016.06.008.

[121] J. M. Hovem, *Marine acoustics: the physics of sound in underwater environments*. Los Altos Hills, California: Peninsula Publishing, 2012, 641 pp., ISBN: 978-0-932146-65-6.

[122] X. Du and F. Hong, *QiandaoEar22: A high quality noise dataset for identifying specific ship from multiple underwater acoustic targets using ship-radiated noise*, May 15, 2024.

[123] Z. Li, S. Xiang, T. Yu, *et al.*, "Oceanship: A large-scale dataset for underwater audio target recognition," in *Advanced Intelligent Computing Technology and Applications*, D.-S. Huang, C. Zhang, and W. Chen, Eds., vol. 14865, Singapore: Springer Nature Singapore, 2024, pp. 475–486, ISBN: 978-981-9755-90-5. DOI: 10.1007/978-981-97-5591-2_40.

[124] P. T. Arveson and D. J. Vendittis, "Radiated noise characteristics of a modern cargo ship," *The Journal of the Acoustical Society of America*, vol. 107, no. 1, pp. 118–129, Jan. 1, 2000. DOI: 10.1121/1.428344.

[125] M. Pricop, T. Pazara, V. Oncica, and D. Atodiresei, "Underwater radiated noise of ships' machinery in shallow water," *Advanced Manufacturing Engineering, Quality and Production Systems*, 2010.

[126] E. H. Roth, V. Schmidt, J. A. Hildebrand, and S. M. Wiggins, "Underwater radiated noise levels of a research icebreaker in the central arctic ocean," *The Journal of the Acoustical Society of America*, vol. 133, no. 4, pp. 1971–1980, Apr. 1, 2013. DOI: 10.1121/1.4790356.

[127] M. F. McKenna, D. Ross, S. M. Wiggins, and J. A. Hildebrand, "Underwater radiated noise from modern commercial ships," *The Journal of the Acoustical Society of America*, vol. 131, no. 1, pp. 92–103, Jan. 1, 2012. DOI: 10.1121/1.3664100.

[128] X. Du and F. Hong, "QiandaoEar22: A high-quality noise dataset for identifying specific ship from multiple underwater acoustic targets using ship-radiated noise," *EURASIP Journal on Advances in Signal Processing*, vol. 2024, no. 1, p. 96, Nov. 8, 2024. DOI: 10.1186/s13634-024-01181-9.

[129] M. Shin, W. Hong, K. Lee, and Y. Choo, "Passive sonar target identification using multiple-measurement sparse bayesian learning," *Sensors*, vol. 22, no. 21, p. 8511, Nov. 4, 2022. DOI: 10.3390/s22218511.

[130] X. Yao, S. Liu, J. Chen, *et al.*, "Underwater acoustic target classification using scattering transform with small sample size," *IEEE Sensors Journal*, pp. 1–1, 2024. DOI: 10.1109/JSEN.2024.3419434.

[131] Q. Yao, Y. Wang, and Y. Yang, "Underwater acoustic target recognition based on hilbert–huang transform and data augmentation," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–19, 2024. DOI: 10.1109/TAES.2024.3417435.

[132] C. Yan, S. Yan, T. Yao, *et al.*, "A lightweight network based on multi-scale asymmetric convolutional neural networks with attention mechanism for ship-radiated noise classification," *Journal of Marine Science and Engineering*, vol. 12, no. 1, p. 130, Jan. 9, 2024. DOI: 10.3390/jmse12010130.

[133] Q. Xu, J. Jiang, K. Xu, *et al.*, "Self-supervised learning-for underwater acoustic signal classification with mixup," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 3530–3542, 2024. DOI: 10.1109/JSTARS.2023.3325921.

[134] J. Tang, W. Gao, E. Ma, X. Sun, and J. Ma, "Deep learning based underwater acoustic target recognition: Introduce a recent temporal 2d modeling method," *Sensors*, vol. 24, no. 5, p. 1633, Mar. 2, 2024. DOI: 10.3390/s24051633.

[135] L. Chen, X. Luo, and H. Zhou, "A ship-radiated noise classification method based on domain knowledge embedding and attention mechanism," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107 320, Jan. 2024. DOI: 10.1016/j.engappai.2023.107320.

[136] L. Chen, X. Luo, and H. Zhou, "A hierarchical underwater acoustic target recognition method based on transformer and transfer learning," in *2024 6th International Conference on Image, Video and Signal Processing*, Ikuta Japan: ACM, Mar. 14, 2024, pp. 16–24, ISBN: 9798400716829. DOI: 10.1145/3655755.3655758.

[137] F. Ahmad, M. Z. Ansari, R. Anwar, B. Shahzad, and A. Ikram, "Deep learning based classification of underwater acoustic signals," *Procedia Computer Science*, vol. 235, pp. 1115–1124, 2024. DOI: 10.1016/j.procs.2024.04.106.

[138] Q. Yao, Y. Wang, and Y. Yang, "Underwater acoustic target recognition based on data augmentation and residual CNN," *Electronics*, vol. 12, no. 5, p. 1206, Mar. 2, 2023. DOI: 10.3390/electronics12051206.

[139] K. Xu, Q. Xu, K. You, *et al.*, "Self-supervised learning–based underwater acoustical signal classification via mask modeling," *The Journal of the Acoustical Society of America*, vol. 154, no. 1, pp. 5–15, Jul. 1, 2023. DOI: 10.1121/10.0019937.

[140] S.-Z. Tian, D.-B. Chen, Y. Fu, and J.-L. Zhou, "Joint learning model for underwater acoustic target recognition," *Knowledge-Based Systems*, vol. 260, p. 110 119, Jan. 2023. DOI: 10.1016/j.knosys.2022.110119.

[141] B. Sun and X. Luo, "Underwater acoustic target recognition based on automatic feature and contrastive coding," *IET Radar, Sonar & Navigation*, vol. 17, no. 8, pp. 1277–1285, Aug. 2023. DOI: 10.1049/rsn2.12418.

[142] J. Li, B. Wang, X. Cui, S. Li, and J. Liu, "Underwater acoustic target recognition based on attention residual network," *Entropy*, vol. 24, no. 11, p. 1657, Nov. 15, 2022. DOI: 10.3390/e24111657.

[143] C. Cheng, "Deep learning for models for underwater ship radiated noise classification," Honours, The University of Sydney, 2024.

[144] S. Shen, H. Yang, J. Li, G. Xu, and M. Sheng, "Auditory inspired convolutional neural networks for ship type classification with raw hydrophone data," *Entropy*, vol. 20, no. 12, p. 990, Dec. 19, 2018. DOI: 10.3390/e20120990.

[145] N. A. Chi, P. Washington, A. Kline, *et al.*, "Classifying autism from crowdsourced semistructured speech recordings: Machine learning model comparison study," *JMIR Pediatrics and Parenting*, vol. 5, no. 2, e35406, Apr. 14, 2022. DOI: 10.2196/35406.

[146] R. Basili, A. Serafini, and A. Stellato, "Classification of musical genre: A machine learning approach.," Jan. 1, 2004.

[147] Y. Zeng, H. Mao, D. Peng, and Z. Yi, "Spectrogram based multi-task audio classification," *Multimedia Tools and Applications*, vol. 78, no. 3, pp. 3705–3722, Feb. 2019. DOI: 10.1007/s11042-017-5539-3.

[148] P. Zhu, Y. Zhang, Y. Huang, C. Zhao, K. Zhao, and F. Zhou, "Underwater acoustic target recognition based on spectrum component analysis of ship radiated noise," *Applied Acoustics*, vol. 211, p. 109 552, Aug. 2023. DOI: 10.1016/j.apacoust.2023.109552.

[149] ZhuPengsen, *ZhuPengsen/method-for-splitting-the-DeepShip-dataset*, Oct. 25, 2024.

[150] P. Zhu, Y. Zhang, Y. Huang, *et al.*, "SFC-sup: Robust two-stage underwater acoustic target recognition method based on supervised contrastive learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–23, 2023. DOI: 10.1109/TGRS.2023.3329653.

[151] B. Lin, L. Gao, P. Zhu, Y. Zhang, and Y. Huang, "An underwater acoustic target recognition method based on iterative short-time fourier transform," *IEEE Sensors Journal*, vol. 24, no. 16, pp. 26 199–26 210, Aug. 15, 2024. DOI: 10.1109/JSEN.2024.3424500.

[152] X. Cao, X. Zhang, R. Togneri, and Y. Yu, "Underwater target classification at greater depths using deep neural network with joint multiple-domain feature," *IET Radar, Sonar & Navigation*, vol. 13, no. 3, pp. 484–491, Mar. 2019. DOI: 10.1049/iet-rsn.2018.5279.

[153] V. E. Premus, M. E. Evans, and P. A. Abbot, "Machine learning-based classification of recreational fishing vessel kinematics from broadband striation patterns," *The Journal of the Acoustical Society of America*, vol. 147, no. 2, EL184–EL188, Feb. 1, 2020. DOI: 10.1121/10.0000774.

[154] A. Dhand, *Thesis-ml*, Dec. 16, 2024.

[155] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy*, vol. 182, pp. 72–81, Sep. 2019. DOI: 10.1016/j.energy.2019.05.230.

[156] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, "A CNN-LSTM-based model to forecast stock prices," *Complexity*, vol. 2020, A. E. I.-B. Hassanien, Ed., pp. 1–10, Nov. 23, 2020. DOI: 10.1155/2020/6622927.

[157] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional CNN-LSTM model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany: Association for Computational Linguistics, 2016, pp. 225–230. DOI: 10.18653/v1/P16-2037.

[158] M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. S. Choi, and B.-W. On, "Fake news stance detection using deep learning architecture (CNN-LSTM)," *IEEE Access*, vol. 8, pp. 156 695–156 706, 2020. DOI: 10.1109/ACCESS.2020.3019735.

[159] R. Vankdothu, M. A. Hameed, and H. Fatima, "A brain tumor identification and classification using deep learning based on CNN-LSTM method," *Computers and Electrical Engineering*, vol. 101, p. 107 960, Jul. 2022. DOI: 10.1016/j.compeleceng.2022.107960.

[160] M. Z. Islam, M. M. Islam, and A. Asraf, "A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using x-ray images," *Informatics in Medicine Unlocked*, vol. 20, p. 100 412, 2020. DOI: 10.1016/j.imu.2020.100412.

[161] Haris Iqbal. "PlotNeuralNet," GitHub. (Nov. 6, 2020).

[162] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980.

[163] gunes. "Answer to "should i normalize all data prior feeding the neural network models?"" Cross Validated. (Apr. 5, 2020).

[164] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, Mar. 31, 2010, pp. 249–256.

[165] Y. Wu and K. He, *Group normalization*, 2018. DOI: 10.48550/ARXIV.1803.08494.

[166] J. Pons, "Deep neural networks for music and audio tagging," Doctorate, Universitat Pompeu Fabra, Barcelona, 2019.

[167] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds., vol. 7700, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–48, ISBN: 978-3-642-35288-1. DOI: 10.1007/978-3-642-35289-8_3.

[168] G. Ruffini, D. Ibañez, M. Castellano, *et al.*, "Deep learning with EEG spectrograms in rapid eye movement behavior disorder," *Frontiers in Neurology*, vol. 10, p. 806, Jul. 30, 2019. DOI: 10.3389/fneur.2019.00806.

[169] Chris Kroenke. "Normalizing spectrograms for deep learning," Chaski. (Aug. 20, 2022).

[170] P. Primus and G. Widmer, *On frequency-wise normalizations for better recording device generalization in audio spectrogram transformers*, Jun. 20, 2023.

[171] N. Simic and A. Gavrovska, "Normalization of audio signals for the needs of machine learning," in *2023 31st Telecommunications Forum (TELFOR)*, Belgrade, Serbia: IEEE, Nov. 21, 2023, pp. 1–4, ISBN: 9798350303131. DOI: 10.1109/TELFOR59449.2023.10372705.

[172] R. J. Hodrick and E. C. Prescott, "Postwar u.s. business cycles: An empirical investigation," *Journal of Money, Credit and Banking*, vol. 29, no. 1, p. 1, Feb. 1997. DOI: 10.2307/2953682.

[173] E. T. Whittaker, "On a new method of graduation," *Proceedings of the Edinburgh Mathematical Society*, vol. 41, pp. 63–75, Feb. 1922. DOI: 10.1017/S0013091500077853.

[174] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 trend filtering," *SIAM Review*, vol. 51, no. 2, pp. 339–360, May 2009. DOI: 10.1137/070690274.

[175] X. Wang, Y. Zhao, X. Teng, and W. Sun, "A stacked convolutional sparse denoising autoencoder model for underwater heterogeneous information data," *Applied Acoustics*, vol. 167, p. 107 391, Oct. 2020. DOI: 10.1016/j.apacoust.2020.107391.

[176] Y. Dong, X. Shen, and H. Wang, "Bidirectional denoising autoencoders-based robust representation learning for underwater acoustic target signal denoising," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–8, 2022. DOI: 10.1109/TIM.2022.3210979.

[177] S. H. Pauline, R. Narayanamoorthi, and S. Dhanalakshmi, "A low-complexity underwater acoustic signal denoising technique based on multi-stage adaptive filter configuration," in *OCEANS 2022 - Chennai*, Chennai, India: IEEE, Feb. 21, 2022, pp. 1–4, ISBN: 978-1-66541-821-8. DOI: 10.1109/OCEANSChennai45887.2022.9775479.

[178] Y. Yin, W. Dai, Z. Zhang, Y. Shi, and S. Sun, "Research on the line spectrum denoising detection based on multi-scale feature autoencoder," *Journal of Physics: Conference Series*, vol. 2517, no. 1, p. 012 006, Jun. 1, 2023. DOI: 10.1088/1742-6596/2517/1/012006.

[179] A. Zhou, W. Zhang, G. Xu, X. Li, K. Deng, and J. Song, "DBSA-net: Dual branch self-attention network for underwater acoustic signal denoising," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1851–1865, 2023. DOI: 10.1109/TASLP.2023.3275030.

[180] Y. Song, F. Liu, and T. Shen, "Method of underwater acoustic signal denoising based on dual-path transformer network," *IEEE Access*, vol. 12, pp. 81 483–81 494, 2024. DOI: 10.1109/ACCESS.2022.3224752.

[181] W. Yang, W. Chang, Z. Song, Y. Zhang, and X. Wang, "Transfer learning for denoising the echolocation clicks of finless porpoise (*Neophocaena phocaenoides sunameri*) using deep convolutional autoencoders," *The Journal of the Acoustical Society of America*, vol. 150, no. 2, pp. 1243–1250, Aug. 1, 2021. DOI: 10.1121/10.0005887.

[182] G. Li, W. Bu, and H. Yang, "Research on noise reduction method for ship radiate noise based on secondary decomposition," *Ocean Engineering*, vol. 268, p. 113 412, Jan. 2023. DOI: 10.1016/j.oceaneng.2022.113412.

[183] R. Gao, M. Liang, H. Dong, X. Luo, and P. N. Suganthan, *Underwater acoustic signal denoising algorithms: A survey of the state-of-the-art*, Jul. 18, 2024. DOI: 10.48550/arXiv.2407.13264.

[184] M. M. Khan, R. H. Ashique, B. N. Liya, M. M. Sajjad, M. A. Rahman, and M. T. H. Amin, "New wavelet thresholding algorithm in dropping ambient noise from underwater acoustic signals," *Journal of Electromagnetic Analysis and Applications*, vol. 07, no. 3, pp. 53–60, 2015. DOI: 10.4236/jemaa.2015.73006.

[185] G. Li, Y. Han, and H. Yang, "A new underwater acoustic signal denoising method based on modified uniform phase empirical mode decomposition, hierarchical amplitude-aware permutation entropy, and optimized improved wavelet threshold denoising," *Ocean Engineering*, vol. 293, p. 116 629, Feb. 2024. DOI: 10.1016/j.oceaneng.2023.116629.

[186] G. Li, W. Bu, and H. Yang, "Noise reduction method for ship radiated noise signal based on modified uniform phase empirical mode decomposition," *Measurement*, vol. 227, p. 114 193, Mar. 2024. DOI: 10.1016/j.measurement.2024.114193.

[187] H. Li, S. Li, J. Sun, B. Huang, J. Zhang, and M. Gao, "Ultrasound signal processing based on joint GWO-VMD wavelet threshold functions," *Measurement*, vol. 226, p. 114 143, Feb. 2024. DOI: 10.1016/j.measurement.2024.114143.

[188] H. Yang, X. Yang, and G. Li, "Dual feature extraction system for ship-radiated noise and its application extension," *Ocean Engineering*, vol. 285, p. 115 352, Oct. 2023. DOI: 10.1016/j.oceaneng.2023.115352.

[189] G. Li, F. Liu, and H. Yang, "Research on feature extraction method of ship radiated noise with k-nearest neighbor mutual information variational mode decomposition, neural network estimation time entropy and self-organizing map neural network," *Measurement*, vol. 199, p. 111 446, Aug. 2022. DOI: 10.1016/j.measurement.2022.111446.

[190] H. Yang, Y. Cheng, and G. Li, "A denoising method for ship radiated noise based on spearman variational mode decomposition, spatial-dependence recurrence sample entropy, improved wavelet threshold denoising, and savitzky-golay filter," *Alexandria Engineering Journal*, vol. 60, no. 3, pp. 3379–3400, Jun. 2021. DOI: 10.1016/j.aej.2021.01.055.

[191] T. A. Smith and J. Rigby, "Underwater radiated noise from marine vessels: A review of noise reduction methods and technology," *Ocean Engineering*, vol. 266, p. 112 863, Dec. 2022. DOI: 10.1016/j.oceaneng.2022.112863.

[192] W. Zhou and J. Li, "Self-noise suppression for AUV without clean data: A noise2noise approach," in *2023 IEEE Underwater Technology (UT)*, Tokyo, Japan: IEEE, Mar. 6, 2023, pp. 1–5, ISBN: 9798350331752. DOI: 10.1109/UT49729.2023.10103424.

[193] N. Alamdari, A. Azarang, and N. Kehtarnavaz, *Improving deep speech denoising by noisy2noisy signal mapping*, Feb. 21, 2020. DOI: 10.48550/ARXIV.1904.12069.

[194] J. Lehtinen, J. Munkberg, J. Hasselgren, *et al.*, *Noise2noise: Learning image restoration without clean data*, Oct. 29, 2018. DOI: 10.48550/ARXIV.1803.04189.

[195] S. Koh, C. S. Chia, and B. A. Tan, "Underwater signal denoising using deep learning approach," in *Global Oceans 2020: Singapore – U.S. Gulf Coast*, Biloxi, MS, USA: IEEE, Oct. 5, 2020, pp. 1–6, ISBN: 978-1-72815-446-6. DOI: 10.1109/IEEECONF38699.2020.9389338.

[196] Y. Liu, H. Zhang, and X. Zhang, "Using shifted real spectrum mask as training target for supervised speech separation," in *Interspeech 2018*, ISCA, Sep. 2, 2018, pp. 1151–1155. DOI: 10.21437/Interspeech.2018-1650.

[197] M. Irfan, J. Zheng, M. Iqbal, and M. H. Arif, "A novel feature extraction model to enhance underwater image classification," in *Intelligent Computing Systems*, C. Brito-Loeza, A. Espinosa-Romero, A. Martin-Gonzalez, and A. Safi, Eds., vol. 1187, Cham: Springer International Publishing, 2020, pp. 78–91, ISBN: 978-3-030-43363-5. DOI: 10.1007/978-3-030-43364-2_8.

[198] N. Adaloglou. "Intuitive explanation of skip connections in deep learning," AI Summer. (Mar. 23, 2020).

[199] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015. DOI: 10.1007/s11263-015-0816-y.

[200] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, Vancouver, BC, Canada: IEEE Comput. Soc, 2001, pp. 416–423, ISBN: 978-0-7695-1143-6. DOI: 10.1109/ICCV.2001.937655.

[201] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification*, 2015. DOI: 10.48550/ARXIV.1502.01852.

[202] A. L. Maas, "Rectifier nonlinearities improve neural network acoustic models," 2013.

[203] X.-J. Mao, C. Shen, and Y.-B. Yang, *Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections*, Sep. 1, 2016. DOI: 10.48550/arXiv.1603.09056.

[204] J. Akeret, S. Seehars, C. Chang, C. Monstein, A. Amara, and A. Refregier, "HIDE & SEEK: End-to-end packages to simulate and process radio survey data," *Astronomy and Computing*, vol. 18, pp. 8–17, Jan. 2017. DOI: 10.1016/j.ascom.2016.11.001.

[205] A. Krull, T.-O. Buchholz, and F. Jug, *Noise2void - learning denoising from single noisy images*, Apr. 5, 2019. DOI: 10.48550/ARXIV.1811.10980.

[206] G. Ashwini and T. Ramashri, "Denoising of COVID-19 CT and chest x-ray images using deep learning techniques for various noises using single image," in *2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*, Karaikal, India: IEEE, May 25, 2023, pp. 1–6, ISBN: 9798350312126. DOI: 10.1109/IConSCEPT57958.2023.10170038.

[207] D. Liu, J. Liu, P. Yuan, and F. Yu, "A lightweight denoising method based on noise2void for x-ray pseudo-color images in x-ray security inspection," in *2022 4th International Conference on Industrial Artificial Intelligence (IAI)*, Shenyang, China: IEEE, Aug. 24, 2022, pp. 1–6, ISBN: 978-1-66545-120-8. DOI: 10.1109/IAI55780.2022.9976566.

[208] S. Kojima, T. Ito, and T. Hayashi, "Denoising using noise2void for low-field magnetic resonance imaging: A phantom study," *Journal of Medical Physics*, vol. 47, no. 4, pp. 387–393, Oct. 2022. DOI: 10.4103/jmp.jmp_71_22.

[209] T.-A. Song, F. Yang, and J. Dutta, "Noise2void: Unsupervised denoising of PET images," *Physics in Medicine & Biology*, vol. 66, no. 21, p. 214 002, Nov. 7, 2021. DOI: 10.1088/1361-6560/ac30a0.

[210] T.-A. Song and J. Dutta, "Noise2void denoising of PET images," in *2020 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, Boston, MA, USA: IEEE, Oct. 31, 2020, pp. 1–2, ISBN: 978-1-72817-693-2. DOI: 10.1109/NSS/MIC42677.2020.9507875.

[211] A. Krull, T. Vicar, and F. Jug, "Probabilistic noise2void: Unsupervised content-aware denoising," Jun. 4, 2019. DOI: 10.48550/ARXIV.1906.00651.

[212] S. Laine, T. Karras, J. Lehtinen, and T. Aila, *High-quality self-supervised deep image denoising*, Oct. 28, 2019. DOI: 10.48550/ARXIV.1901.10277.

[213] E. Höck, T.-O. Buchholz, A. Brachmann, F. Jug, and A. Freytag, *N2v2 – fixing noise2void checkerboard artifacts with modified sampling strategies and a tweaked network architecture*, Nov. 21, 2022. DOI: 10.48550/ARXIV.2211.08512.

[214] D. Zhang, F. Zhou, F. Albu, *et al.*, *Unleashing the power of self-supervised image denoising: A comprehensive review*, Mar. 25, 2024. DOI: 10.48550/ARXIV.2308.00247.