NTNU

Norwegian University of
Science and Technology

Classification of Marine Vessels using Sonar Data and a Neural Network

Håkon Gimse

Master of Science in Computer Science
Submission date: April 2017
Supervisor: Helge Langseth, IDI

Norwegian University of Science and Technology
Department of Computer Science

Håkon Gimse

Classification of Marine Vessels using Sonar Data and a Neural Network

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

A submarine navigator have to keep track of surrounding ships in order to avoid collision and to gain a tactical advantage. This is currently done manually by a sonar operator, trained to listen through the water and identify ship-types by the sound they emit.

This project presents a review and implementation of different solutions to the problem of audio classification. The goal of this work was to build a system capable of helping submarine navigators identify surrounding obstacles and ships based on the sound recorded by the sonar on board the submerged submarine.

The research aims to uncover the best combination of techniques that can be used for this classification task. This project concerns both the field of signal analysis and artificial intelligence as the system comprises of two parts. The first being a method of extracting informative features from sonar data captured by the submarine. The second part is to feed the processed data into a neural network (NN) and provide a classification of the ship's type.

In this project a system have been developed in order to experiment with a variety of feature extraction techniques and neural network structures to find a solution suitable for the submarine sonar classification problem. The system have been able to place 97.3% of the ships in the correct category when using the highest scoring combination of a feature extraction technique and a neural network. The best found combination was the Mel Frequency Cepstral Coefficients feature extraction technique and a standard feed-forward neural network.

Sammendrag

En ubåt må alltid ha oversikt over alle skip i nærheten for å unngå kollisjon og for å holde en taktisk fordel fra dypet. Dette er nå gjort manuelt av en sonar operatør som er trent opp til å lytte etter lyder i vannet og identifisere forskjellige typer fartøy basert på lyden de lager.

I dette prosjektet presenters forskjellige metoder for klassifisering av lyd-data. Målet med dette arbeidet er å lage et system som kan assistere navigatøren om bord på en ubåt med å identifisere omringende hindringer og skip basert på sonar data.

Forskningen tar sikte på å avdekke den beste kombinasjonen av teknikker som kan løse denne klassifiseringsoppgaven. Prosjektet omhandler både fagfeltet signal-analyse og kunstig intelligens da systemet er bygget opp av to deler. Den første, en metode for å hente ut informative attributter fra rådata fra sonaren på ubåten. Den andre delen i systemet er et nevralt netverk som mates med attributtene funnet i den første delen. Dette netverket kan etter at det er trent opp brukes til klassifisering av ulike skipstyper.

Systemet er designet for å kunne teste en rekke forskjellige kombinasjoner av preprosessering og nevrale netverk for å finne en løsning som passer til å klassifisere sonar-data til skipstyper. Systemet kan klassifisere riktig lyd til riktig fartøy i 97.3% av tilfellene når det kjøres med den konfigurasjonen som har oppnådd den høyeste klassifiserings-treffsikkerheten. Den beste kombinasjonen funnet var en preprosesseringsteknikk kalt Mel Frequency Cepstral Coefficients og et standard nevralt netverk

Preface

This thesis was written for the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU).

The project has been planned in cooperation with Kongsberg Defence & Aerospace AS (KDA), and is a continuation of the preliminary study concerning the same problem. KDA have developed the sonar system currently on board the Norwegian Navy's submarines, and have access to simulation software that can be used to create training data.

The problem to be solved was thought of by the author after spending the compulsory military service on a submarine as a sonar-operator.

The readers of this study are assumed to have a background in computer science.

Acknowledgment

I would like to acknowledge the support from the faculty IDI at NTNU. In particular my supervisor Helge Langseth, who helped me make this project possible by being an invaluable resource person in the field of artificial intelligence.

Kongsberg Defence & Aerospace AS and my contact there, Simen Tronrud has helped me solve the data gathering challenge and granted me access to their simulation software the following semester. Without this prospect of actual data this project would not have been possible.

Håkon Gimse
Trondheim, March 26, 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This work explores how to use artificial intelligence to recognize patterns in sound. The goal was to create a prototype of a system that could be used on board a submarine to assist the sonar operator. In order to build such system both the field of artificial intelligence and the field of signal processing will be studied, as the system will have to use state-of-the-art techniques from both fields. In this chapter the background and motivation for doing this will be discussed, followed by a presentation of the goals of this project with a brief introduction to the scientific approach taken. Finally the structure of the remaining project will be outlined.

1.1 Background and Motivation

A submerged submarine is highly dependent on information about surrounding ships to have a tactical advantage. It has to remain silent, so it can not send out an active sonar signal to search for nearby ships. It has to stay submerged and passively listen for the other ship's acoustic signature. This signature is mostly created by the cavitation sound from the propellers of the ship. This sound travels through water and is intercepted by the submarine. It is essential to know what kind of ship the signal comes from. A fishing boat using a trawl net would for example be crucial to avoid. Interpreting sound signals into classes is a task well suited for a neural network as the signals are very fuzzy and contains a lot of noise. A neural network would be able to provide support to the manual operator in this very crucial task.

The system developed in this project will work as a real time support system to the sonar operator. The system will receive the raw acoustic data from the sonar and be able to suggest classes of surrounding ships. A manual sonar operator will

only be able to listen in one direction at the time, where the proposed system could observe every angle at once. It will also be able to alert the operator when it believes it has classified a nearby ship correctly. The operator could then either confirm or refute this classification, and the system could use this to learn the different classes further.

The solution to this problem is a system that is using raw acoustic data as input, followed by processing and feature extraction and finally returning a class. This ideal solution can be divided into two sub-systems, where the output of the first is the input of the second. This is illustrated in Figure 1.1. This is a more modular approach. A solution to one of the problems can be implemented independently of the solution used in the other and they can still work together.

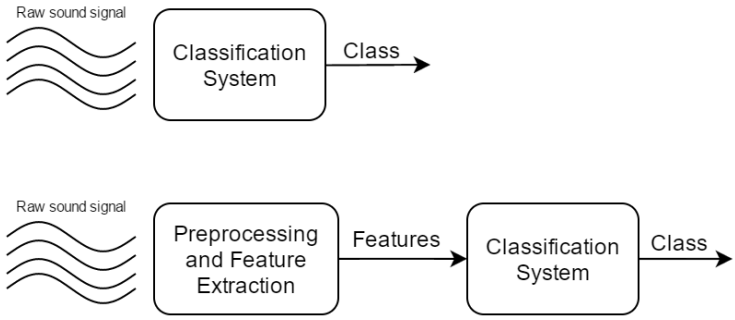

Figure 1.1: A figure showing how the problem can be divided into two sub-problems

The first problem, the preprocessing and feature extraction from raw audio data aims to find a more informative representation of the data, either by extracting key elements in the data or finding new information by analysis. Often the main goal of this task is to find a smaller data set to consider during classification. This will decrease the complexity of the classification task severely.

There is a large community devoted to this field, and countless papers published on the subject. The data this project addresses is different than what most studies about sound processing tend to use. The most obvious difference being that sound travels at a different rate in water than in air. This study aims to answer whether the same techniques used to extract features from environmental sound on land can be used on data sampled under water.

The preprocessing techniques also have to be robust to noise. The ocean contains an abundance of biological noise from whales, dolphins and lesser life forms. It is also a known issue that the ocean between the sound emitting ship, and the submarine often contain different layers of water with different salt concentration,

or temperature. The sound passing through will be disrupted and changed by this.

The second problem regards the classification of the features produced in the first module. The complexity of the neural classifier have to match the complexity of the data to be classified. If the data has features that always lead to a certain classification, this task would be easy, but it is likely that the data has features behaving differently in every data sample. This is a common problem that often renders other traditional algorithms for classification useless. However deep neural networks have proven that they can be able to identify advanced patterns, even in noisy data sets.

1.2 Goals and Research Questions

An unambiguous definition of the problem is given as follows; a manual sonar-operator is not able to accurately and quickly classify all surrounding marine vessels. How can this be improved? A solution to this problem will be of great interest to the navigators of the submarine. The system developed in this project aims to answer whether this problem can be solved with the use of a neural network, and if it can, then what kind of preprocessing techniques and network configurations would be optimal.

The stated goal is defined as follows:

Goal Develop a system where a signal preprocessor and a neural network can work together to classify marine vessels based on their acoustic signature.

The goal is to implement state-of-the-art techniques in each of these fields suitable for the task at hand. The end result will be a system suited for experimentation in order to uncover the potential of such a system. This goal will be approached by researching these fields using a structured literature review, then to implement the techniques according to the state-of-the-art research found in the literature review. The following four research questions are defined as follows:

RQ1 What are the existing solutions for classification of continuous audio-data using neural networks?

RQ2 How are the solutions found in **RQ1** processing the data before feeding it to the neural network?

RQ3 How can the findings be used when creating a classification-system in a new environment?

RQ4 How will different solutions, or combinations of solutions, affect the overall performance of the system?

1.3 Research Method

The approach that will be used to achieve this goal is to use the thorough scientific literature review conducted in the specialization project by the author of this thesis, Gimse [2016]. The review contains an systematic overview of relevant research to the project and is a study and preparation of this project. The focus here is researching the theoretical state-of-the-art in the two fields, to ensure that the techniques found are of high quality in their respective fields. The most promising techniques will be implemented and tested in order to answer the research questions. The bibliography used here will be highly relevant to this project, but left out as it is referenced in the specialization project.

1.4 Thesis Structure

The rest of the report is organized as follows. In Chapter 2 an introduction to the field will be given, covering the theory needed to understand the how and why of the found solutions. Then there is a chapter on how solutions to this problem have been implemented followed by Chapter 4 where the solutions implemented are tested and results are presented. Finally a conclusion is presented in Chapter 5. This chapter also includes the recommendations for further work. The goal of this project is ultimately to build a system where feature extraction and neural network techniques can be tested in combination, and to prove that the task at hand can be solved by such a system.

Chapter 2

Background Theory

This chapter provides insight into the field of audio analysis and neural networks, and the techniques this project is concerning. This chapter also contains a more technical description of the sonar system, the data sampled and the domain this project will focus on, namely underwater environmental sounds.

2.1 Sonar Data

Sonar was originally an acronym for Sound Navigation and Ranging. This is a technique where sound propagation is used to retrieve information about the surrounding area. It is most commonly used with submerged submarines, either as a means of communication, navigation or listening for other vessels on or under the surface of the water.

Sonar systems are divided into two groups, passive sonar and active sonar. An active sonar is a system that actively tries to find objects by emitting a strong pulse into the water and gathering information from the echo signal bouncing back from said objects. This is a commonly used method of finding hidden submerged submarines. The searching ship is looking for areas where the echo signal is returning too fast to have met the bottom of the ocean, and assumes this to be because the signals are bouncing back off a submarine. Before the introduction of radar, acoustic echolocation was used in air as well.

A passive sonar on the other hand is a system that listens without emitting a signal. It is dependant on other vessels making noise in the water. Typically this is used by hiding submarines, as emitting an active sonar signal will surely give away the location of the source of such sound. By using a passive sonar system the submarine can get an overview of surrounding vessels without giving away its location. The most informative sounds captured by the passive sonar is typically

the cavitation frequency of a nearby vessel. This is unique for most ships and can yield information about the size, speed and type of the ship. Intermittent sound sources can also be heard, e.g. a flushing toilet or a wrench hitting the hull of the ship. A passive sonar will be used in this project, as this is the type the Norwegian Navy is using.

There are several performance factors to take into account when using a sonar for navigation. The speed of sound is the most important as this approach is based on the propagation of the sound. Sound travels more slowly in freshwater than in saltwater, and even in saltwater there is great differences determined by the water's bulk modulus and mass density. Bulk modulus is a measure of a substance's resistance to uniform compression, here this is determined by the temperature, pressure and salinity. Salinity is the saltiness of a body of water. The affect given by the mass density of water is relatively small compared to that of the bulk modulus. The speed of sound in the air is approximately 340 m/s, while in saltwater it is approximately 1500 m/s. The formula used to calculate the precise speed of sound in water is presented by The National Physical Laboratory [2005] in Equation 2.1 where D is the depth given in kilometers, S is salinity in parts per thousand and $t = T/10$ where T = temperature in degrees Celsius. $c(D, S, t)$ is the speed of sound at depth D, and $c(0, S, t)$ is the speed of sound on the surface, given by Equation 2.2.

$$c(D,S,t) = c(0,S,t) + (16.23 + 0.253t)D + (0.213 - 0.1t)D^2 + [0.016 + 0.0002(S-35)] * (S-35)tD \tag{2.1}$$

$$c(0,S,t) = 1449.05 + 45.7t - 5.21t^2 + 0.23t^3 + (1.333 - 0.126t + 0.009t^2)(S-35) \tag{2.2}$$

In order to determine the direction of a sound source, a sonar is built using a circular array of hydrophones. These are microphones designed to work under water. Each of these captures the environmental sounds in the water, and the direction of a sound source is determined based on the position in the circle of the hydrophones capturing the strongest signal.

2.1.1 Cavitation

A propeller absorbs the torque from the ships motor at given revolutions. In turn the propeller converts this to thrust, driving the ship through the water. The International Institute of Marine Surveying [2015] states that according to Bernoulli's law the passage of a hydrofoil (propeller blade section) through the water causes a positive pressure on the face of the blade and a negative pressure on

its back. It is the resolution of the pressures that results in the torque requirement and the thrust development of the propeller. The negative pressure causes any gas in solution in the water to evolve into bubbles and when these collapse a sound is emitted. The repetition of such sound create an acoustic signature unique to most ships.

2.2 Signal Processing

A sound is an interpretation done by the brain of waves of pressure traveling through a medium. These waves are produced by any vibrating object, like the vocal cords of a human, or the cavitation from a ship. Waves come in different frequencies, determined by the frequency of the vibrating object and the medium the sound travels through. The human ear can only perceive frequencies between 20 and 20000 Hz. Even when the human ear can not hear a sound, we can use a microphone to record it, digitize it and then analyse it.

A microphone is simply a device that records pressure variations and transform this into a voltage signal. To be able to use this data in analysis, we have to transform this analogue data into discrete data. This is done by a technique called sampling. This is done by measuring and storing certain values in the analogue data at specific points in time. The frequency of such points is determined by the sampling rate. The highers sampling rate, the more data points is gathered, and the resulting data curve becomes a closer approximation of the original analogue signal. An illustration of this is shown in Figure 2.1. Note how variations in the original signal is lost between each sampling point. When the signal to be recorded has higher frequencies, even more information is lost when the sample rate is low.

A theorem describing the relationship between the sample rate and the frequency measured is described by National Instruments [2015]. This theorem is called the Nyquist Sampling Theorem, and states that the sampling rate f_s must be greater than twice the highest frequency component of interest in the measured signal. This frequency is often referred to as the Nyquist frequency, f_N. This is shown mathematically in Equation 2.3 and the reason why this theorem is needed is illustrated in Figure 2.2. The figure demonstrates how a sampling rate of twice the frequency of interest guarantees that the main shape of the signal is accurately reconstructed. Usually the sampling rate is much higher to achieve a smoother digital signal. When these discrete data values are gathered, analysis can be computerized.

$$f_s > 2 * f_n \tag{2.3}$$

We use signal processing to manipulate a signal to change its characteristics

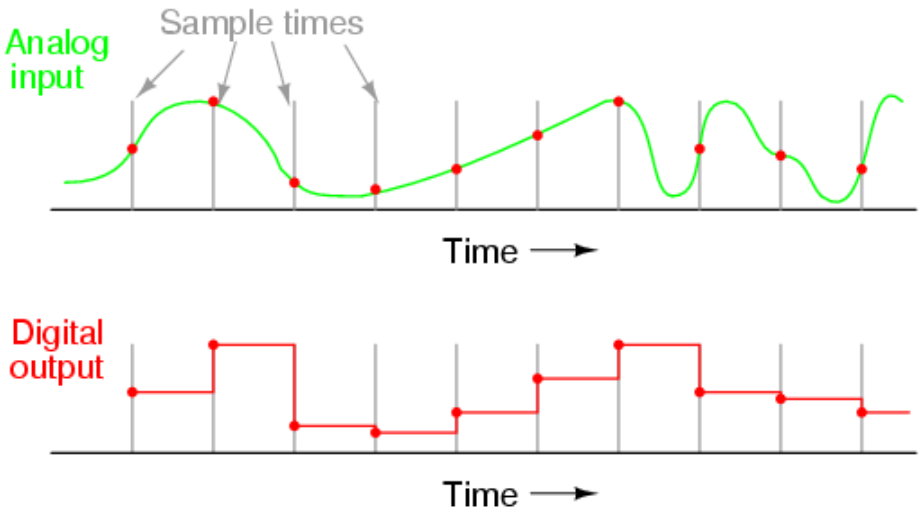

Figure 2.1: Sampling an analogue signal into a digital signal.

or to extract additional information. The problems signal processing typically aims to solve is; noise reduction, correcting distortion and extraction of indirect quantities in measured signals. It is this last one that is most relevant to this project as a sonar is used to capture data, and from this data information about a ship is extracted. The type, distance and speed of a ship are examples of indirect quantities in this signal.

2.2.1 Time-frequency domain

A sound signal is generally plotted in the time domain. This way it is easy to study how the signals amplitude is changing over time. However, many signals of interest have changing frequency characteristics, e.g. speech. To capture this trait it is helpful to study a signal, plotted in the frequency domain. This is a representation of the signal that shows how much of the signal lies within each given frequency. These two representations are presented in Figure 2.3. A methodology called time-frequency analysis is commonly used in signal processing. This is a group of methods where the signal is studied both in the time domain and in the frequency domain simultaneously. To find a representation of a signal in the frequency domain a Fourier transform is typically used. The most basic version of this is the short-time Fourier transform, but more sophisticated methods like

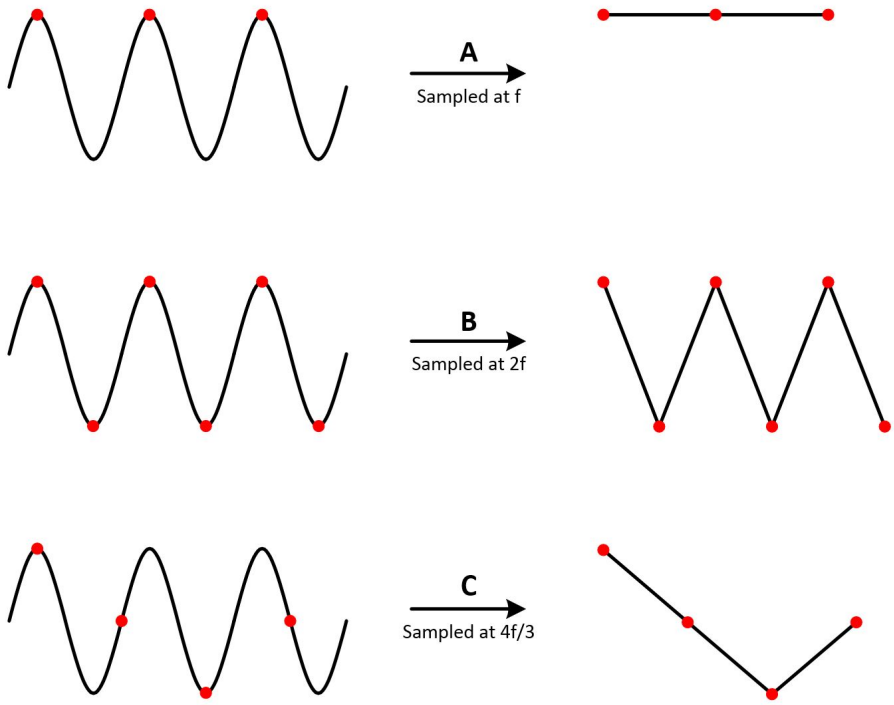

Figure 2.2: National Instruments [2015] demonstrates how a low sampling rate will lead to an inaccurate digital approximation of a signal.

wavelets are also commonly used.

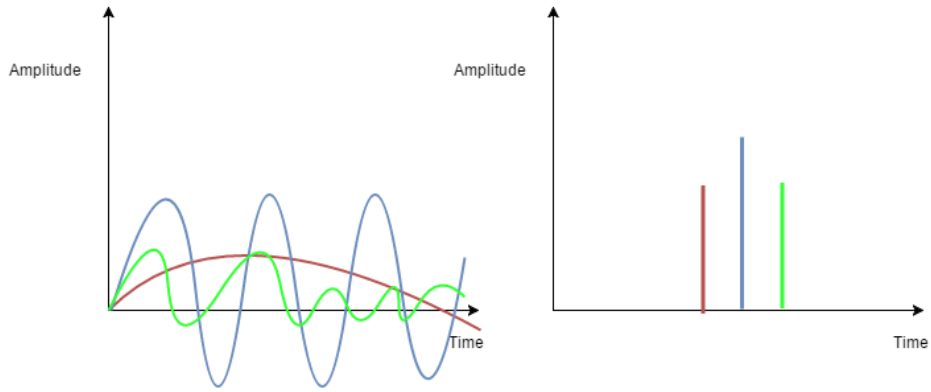

Figure 2.3: An illustration of how a signal is represented in the time domain (right) and the frequency domain (left). Each yielding different information.

2.2.2 Short-time Fourier Transform

All signals can be represented as the sum of sinusoidal signals. By identifying the key sinusoidal signals one can selectively remove or keep those desired, i.e. if a noise source is known with its sinusoidal, it can be isolated and removed. The short-time Fourier transform (STFT) is a way of identifying such sinusoidal signals as the signal's parameters change over time. This is done by dividing a signal into short time segments and doing a Fourier transform on each of these. Doing this results in coefficients for each segment. These coefficients are used as a frequency domain representation of each of these short time segments. By piecing these representations back together in the time domain it is possible to study the signal both in time and in frequency.

2.2.3 Wavelet Transform

Wavelet transform is as mentioned a more sophisticated way of analysing a signal in both domains. The main difference between the two approaches is that wavelet transform approximates a signal using short waves called wavelets instead of sinusoidal signals. Where the Fourier transform returns coefficients for each sinusoidal signal found, the wavelet transform returns two values for each

wavelet found, namely the scale and the translation of the wavelet. The translation decides the timing of how the wavelet translates through the signal and the scale decides the amplitude of the wavelet. The advantage of using wavelets over sinusoidal functions is that they are short, and therefore more suitable when a longer signal is divided into short segments as more wavelets can be used each segment. The Fourier transformation using continuous sinusoidal signals always have to cut these short at the cost of resolution in the approximation.

2.2.4 Mel-frequency Cepstral Coefficients

Mel-frequency cepstral coefficients (MFCC) is a another advanced time-frequency analysis technique where the resulting features are the coefficients of amplitudes in a transformed spectrum. This is commonly known as the most popular technique in the field of speech recognition. Why that is will become apparent, when the steps are described in the following paragraph. The steps for finding the MFCC is as follows

1. Transform the signal into the frequency domain. This is typically done with Fourier transform on segmented data like in STFT
2. Map the frequency domain representation found in the first step into the mel scale described below.
3. The logarithmic value of each of these mel frequencies are calculated.
4. The logarithmic values found in step 3 is interpreted as a signal and the discrete cosine transform is done on this signal. This transform is practically a Fourier transform, but only using cosine functions to approximate the signal.
5. A spectrum of the transformed signal is produced named a mel-frequency cepstrum. The coefficients extracted are the values of the amplitudes in this spectrum.

The mel scale is a scale created to measure the perceived pitch instead of the actual pitch. This was a solution to the human ear's inability to differentiate correctly between pitches in the highest bands of hearing. The mel scale describes the human auditory system on a linear scale and here we can express the difference in pitches we perceive. The relationship between the mel scale and the frequency scale is shown in Figure 2.5.

A spectrogram is a visualization of the spectrum of frequencies in a signal as they vary over time. An example of this is given in Figure 2.4.

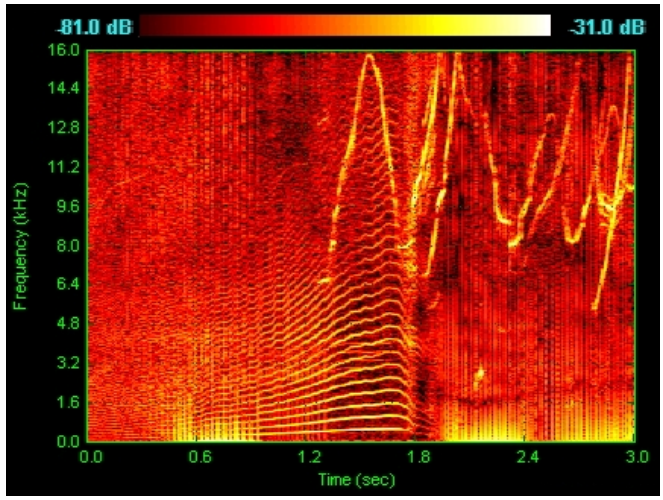

Figure 2.4: A spectrogram showing the vocalizations of a dolphin. This figure also show the power of the sound as a heat map.

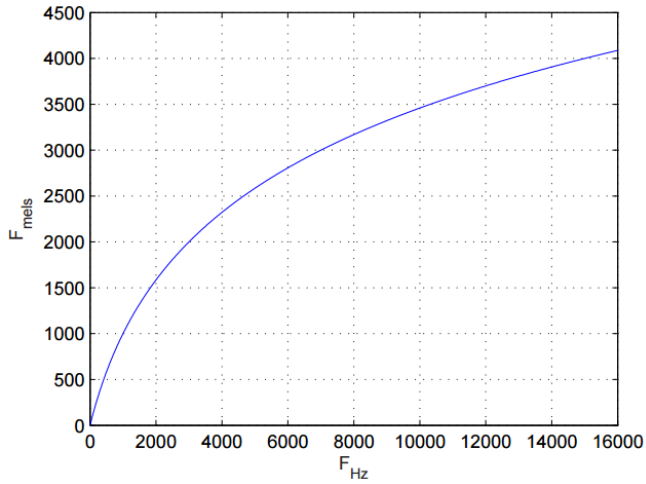

Figure 2.5: The mel scale. A scale based on the perceived pitch.

2.2.5 Spectral Density Estimation

Here a statistical approach is taken to estimate the spectral density of an unsteady signal. The spectral density describes the distribution of power in a signal. Doing this is also a feature extraction technique, where it is possible to learn patterns like periodicity in the data. This alone can be a very informative feature in seemingly random data.

2.3 Artificial Neural Networks

A neural network is a computational model used to approximate functions that depend on a large number of inputs. It is inspired by the central nervous system in the human brain where neurons are connected by synapses to form a network. Information is flowing through the brain by electrical impulses. Each neuron does some kind of processing on the signal flowing through the network, resulting in a different output than input to the network. This network is trained over time to do the desired computation. In artificial neural networks this is mimicked by connecting a large amount of nodes together to form a network. Each node has a function in order to change the incoming signal. The nodes are grouped together to form layers. A neural Network has one input layer, one output layer and at least one hidden layer. The hidden layers are the nodes doing the actual computation how this is done is described in the next Section 2.3.1. The weight of each connection is changed in order to train the network to approximate a target function.

A neural network is trained using supervised learning. Typically this is done by feeding the network with training data labeled with the correct class, than the network does the classification and adjusts its configuration based on the error in the classification. With enough training data the network becomes able to generalize beyond the training data and learn to classify data never seen in the training set. Exactly how this training is done will be explained in Section 2.3.2.

2.3.1 Activation Functions

The activation function is the function deciding the output from each node in a neural network given the input. This can be a simple function like an AND function, only passing a activation signal on if all the incoming signals are high, or it can be more complex. Some of the popular activation functions suggested by Nielsen [2015] are shown in Table 2.1. A widely used input composition for activation functions is the nonlinear weighted sum of signals and weights. This is given by the Formula 2.4 where w is connected weights and x is the output value of the node connected by weight w. These values are often represented as a

weight matrix and an input matrix for a speed advantage as matrix multiplication can be computationally inexpensive.

$$y_i = \sum_i w_i x_i \tag{2.4}$$

Name	Equation
TanH	$f(x) = \frac{2}{1+e^{-2x}} - 1$
Sigmoid a.k.a logistic or soft step	$f(x) = \frac{1}{1+e^{-x}}$
Rectified Linear Unit (ReLU)	$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$
Sinusoid	$f(x) = sin(x)$

Table 2.1: A selection of activation functions and their equation.

2.3.2 Training

To be able to learn a concept the neural network has to be trained to do so. Training a neural network means tweaking parameters until it yields the desired results. The parameters that can be changed depends on the type of network, but mostly we talk about changing the weights of the connections in a network. To to so would change the influence one node has on the node on the other side of the connection. The goal is to make a network so that it understands what connections usually supplies it with information that corresponds with the class, and prioritize these. This way the neural network is able to learn patterns the programmer did not explicitly introduced.

An example of an algorithm for training a neural network is the backpropagation algorithm. This algorithm calculates the error in the classification in each training example by comparing the prediction given by the network and the label of the example. Then a gradient of a loss function is calculated with respect to the weights in the network. This error gradient is than propagated back through the network and the weights are updated in order to minimize the loss.

2.3.3 Overfitting

Overfitting is a term used to describe a network when it becomes too specifically fitted to the training data. The aim of a predicative model like a neural network is to train on a diverse data set from each class in order to gain a general concept

of each class. When this is done right the model can predict data it have never seen before correctly even if it varies from the training data in the same class. If the model fits too good to the training data to be able to generalize, it is called overfitting.

2.3.4 Standard Feed-forward Neural Network

The feed-forward neural network is the most simple form of neural network. It is a neural network wherein connections between two nodes does not form a cycle. The data flows simply from the input layer, iteratively through the hidden layers and ends up in the output layer. This is illustrated in Figure 2.6. Note how all edges are directed in one direction, from the input to the output layer.

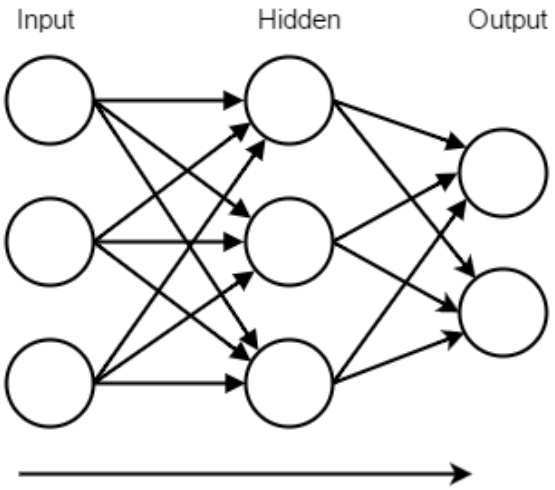

Figure 2.6: A standard feed-forward neural network

2.3.5 Convolutional Neural Networks

A convolutional neural network is a type of feed-forward network, but the layers are not fully connected, that is every node in each layer is only connected to a subset of the nodes in the next layer. Where a standard feed-forward network has individual connections between the layers, the convolutional network has a set of connections from one node in the incoming layer to n nodes in the next, where n is a number between 1 and the total number of nodes in the next layer. This set of connections are called a filter, and are reused throughout the layer from each

node to the next layer. This filter is used iteratively through the incoming nodes, by changing the start- and end-point of the connections, to create new signals. This is shown in Figure 2.7 presented by Nielsen [2015]. The configuration of these filters are being trained to capture new informative features.

It can also be useful to use several different filters between the same layers, as more features can be identified. Using several layers and the sliding technique often lead to a huge expansion in nodes in the following layer behind these filters. This is illustrated in Figure 2.8, where three different filters are used. Too many nodes in a layer can be a problem, as the computational cost of training increases. To solve this problem we use a technique called pooling to reduce the dimensionality of the data. This is done by pooling together a number of nodes in the incoming layer, hence the name, and extracting some single value from them to pass on. A popular example is max-pooling. Here the highest activation value in the pool is simply passed on and all the other values are disregarded. This is illustrated by Britz [2015], here shown in Figure 2.9 where each number on the left is the activation level of a node, and the number of corresponding color on the right is the resulting activation after the pooling layer.

The convolutional network methodology has recently proved very useful for complex classification task like image recognition. A testament to this is The ImageNet Large Scale Visual Recognition Challenge held every year. It is a competition in image recognition where several teams train their algorithms on 1.2 million images in 1,000 categories. An algorithm has to have the correct class in the top five predictions to be correct. Russakovsky et al. [2015] is presenting results where almost every high ranking algorithm used is based on convolutional neural networks. They also show that the increase every year in the best algorithm is very high (between 3 and 5 percent). This shows that this is a fast moving field.

2.3.6 Recurrent Neural Networks

A recurrent neural network is contradictory to a feed-forward network, a network where connections between nodes form a cycle. This allows the network to pass information back through the network. Typically the information passed is the state of the node, which is dependant on signal previously processed. This way the recurrent network achieves a form of memory. This makes them useful when classifying data that is sequential, that is when the order of the data matters. An example of this is speech, where the first words could affect the probability of what the next would be. A recurrent neural network is illustrated in Figure 2.10, where the red arrows denote the connection where the recurrence happens.

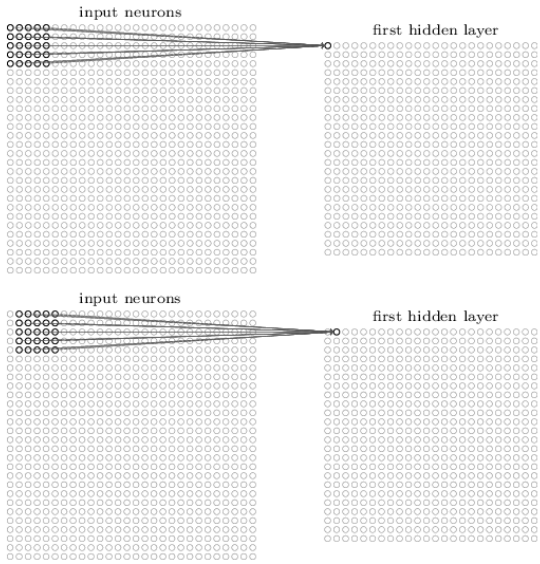

Figure 2.7: A single matrix of weights, or filter, is used to map several input nodes to a single hidden node. Here the filter is moved one step to the right. It will continue using the same weights throughout the entire input layer.

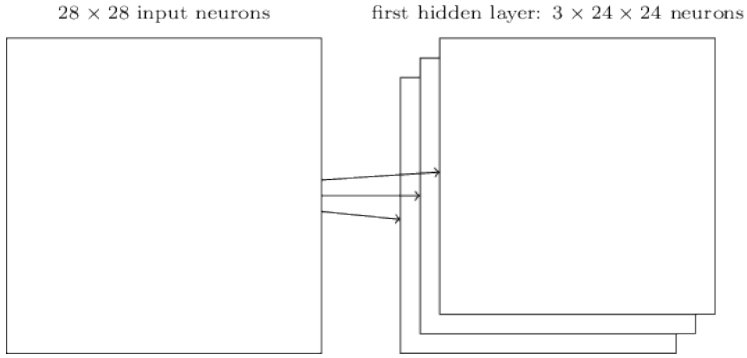

Figure 2.8: The expansion of nodes becomes apparent when several filters are used. Here three different filters are used on a layer with 28 input nodes.

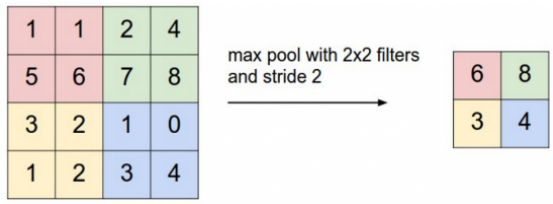

Figure 2.9: A layer where features are condensed based on the pooling function, here max-pooling

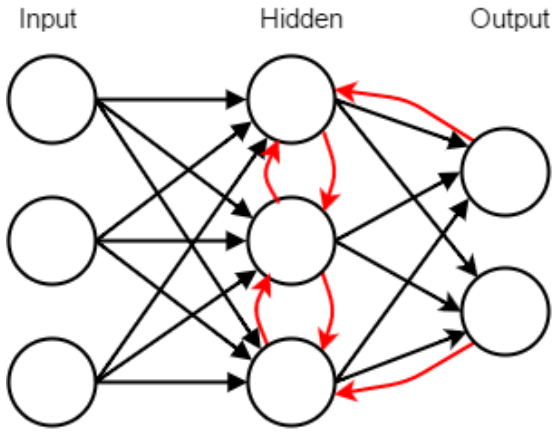

Figure 2.10: A recurrent neural network. The connections that create the recurrence are denoted as red arrows.

Chapter 3

Architecture and Implementation

In this chapter the architecture and implementation of the classification system is presented. The system incorporates several techniques of learning with neural networks, and several techniques concerning feature extraction. Each execution of the system will combine a feature extraction technique and a neural network configuration. The combination of the two is selected in a simple graphical user interface.

The different components of the system will be discussed in detail in the following sections. This will mostly remain conceptual, and readers interested in the source code can download the project from: `https://github.com/hakon0601/MastersProject`.

3.1 Architecture

An illustration of the systems architecture is presented in Figure 3.1. The system has a module-based architecture, where several independent modules form the system. This architecture allows further work with the system as several different techniques can be developed and used with the system without having to rewrite any of the old modules.

3.2 Data loader

The first module in the system is the data loader. This is a module where data is loaded into the system and organized in a specific manner, in order to be passed

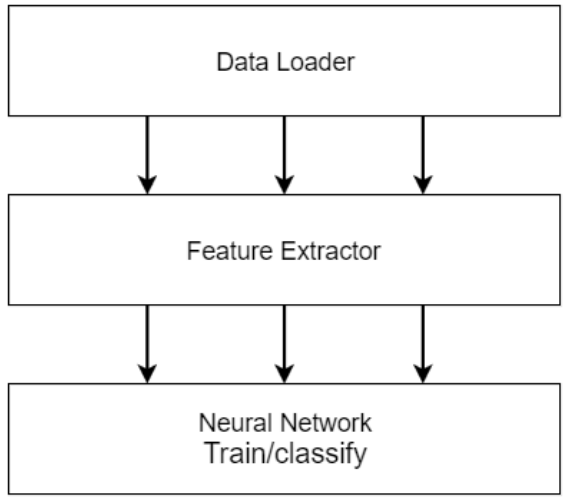

Figure 3.1: The module based architecture of the system developed in this work. Arrows denote the data flow and require a specific format. A new module would have to comply with this format for be used with the rest of the system.

to and used by the next module. The data loader module could easily be replaced by a similar module where continuous audio data is used instead of sound files. If the system deploys on board a submarine this would be more natural as sound is recorded continuously.

3.2.1 Data

Raw audio files are the base form of the data used in this work. This data have been gathered using a sonar simulation system developed by KDA. This acoustical tool allowed creation of a variety of ships with customizable engine configurations. Ships could be placed anywhere around the submarine and their relative sound to the submarine were generated. A total number of 85 sound files of 10 seconds duration each were gathered and used in this project. A full overview of the different configurations used in the data set can be found here: `http://folk.ntnu.no/haakongi/Sonar_Data_TSUS_Public.xlsx`. As seen here a number of 7 different ship classes were selected and generated in different scenarios.

3.2.2 Data expansion

As discussed in Chapter 2 a neural networks ability to learn is highly dependant on the size and quality of the data set used in training. To expand the data set further the data was sampled into several shorter samples. A 10 second recording could be sampled down to 100 unique recordings of 1 second each, where each sample have some overlap from the previous and next sample. This is configurable in the system created and options to set the number of samples per file and sample length have been created. In addition to this the system has an option to create samples with some random noise for each sample. This technique allows the data set to be multiplied several times.

3.3 Feature extractor implementation

The main goal here is to process the raw data into a more efficient representation without loss of key information to the classification procedure. The system developed in this project incorporates several feature extraction techniques. The options available are listed in Table 3.1. Each is discussed individually in Chapter 2. One of these are selected for each run in the system. The first four produce a 1-dimensional vector of features extracted from each sample. This is suitable for the standard feed-forward neural network and the recurrent neural network as they require a 1-dimensional vector as input. The last option, spectrogram, is different and created specially in compliance with the requirements of the convolutional neural network. This feature extraction technique creates a 2-dimensional plot of the time-frequency domain and feeds it to the convolutional neural network as an image.

All third party libraries used and mentioned here will be discussed in Section 3.5.

Feature Extraction Technique
Mel Frequency Cepstral Coefficients
Short-Time Fourier Transformation
Spectral Density Estimation
Wavelet Transformation
Spectrogram

Table 3.1: Available feature extraction techniques in the classification system.

Mel Frequency Cepstral Coefficients

The implementation of the MFCC algorithm described in Chapter 2 is done by using Librosa, a third party library for audio and music analysis. The magnitude and phase of each frequency bin at each time frame is computed and returned to the neural network. The number of coefficients computed can be selected by the user during run-time of the system. An increase in the number of coefficients will give a more accurate representation of the raw data in the same way that a higher sampling rate will. Finding the ideal number of coefficients to use with a data set can only be done by experimentation. This experimentation is presented in Chapter 4. The next chapter presents experiments and findings about a variety of feature extraction techniques and parameters.

Short-Time Fourier Transformation

The short-time Fourier transformation is similar to the MFCC technique. The STFT method provided by Librosa accepts parameters and the window size in the transformation have been exposed in the system and can be set by the user. This allows the user to test configurations where the data resolution is higher.

Spectral Density Estimation

The spectral density estimation is done using the SciPy library discussed in Section 3.5. This package offers an implementation of Welch's Method presented by Welch [1967], used to compute an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms. The result is a highly compressed feature vector.

Wavelet Transformation

To compute the wavelet transformation of a signal, the library PyWavelets is used. The Discrete Wavelet Transform is computed and the approximation and detail coefficients for each sample is extracted as the resulting features.

Spectrogram

A spectrogram is presented on image form, or as a 2-dimensional vector. This makes it suitable to use image recognition techniques as convolutional neural networks often do. The spectrogram extractor produces an image using the 1-dimensional data sample and this image is used by the neural network. The

library Matplotlib is used to do the necessary computation. The system also incorporates an option to save spectrogram's as image files. This was implemented in order to see if humans were able to differ between the 7 classes, when given only the spectrogram.

3.4 Neural Network implementation

This goal of this project is to find the most suitable neural network scheme for the task at hand. Three neural network types have been implemented and each of these facilitate experimentation with network configuration and parameters such as network depth, bias, activation functions etc. Each individual network implementation will be discussed further in this section.

The neural networks created in this project all have the same core functionality. They all inherit an abstract base neural networking class and are all made to implement the same three methods: construction of the neural network, training of the neural network, and lastly testing the performance of the trained neural network.

The neural networks have been implemented using Tensorflow described in Section 3.5.

Standard Feed-forward Neural Network

The construction of the network structure is done by defining nodes and edges in the Tensorflow data flow graph as discussed in Section 3.5. A node in the graph is a variable, in Tensorflow called a Tensor, of a predefined size and type. The first thing needed in a neural network data flow graph is a way of receiving input data. This is done by creating a Tensor with the size and data type of the output from the feature extractor. Therefore the neural network structure is defined only after the feature extractor has processed the data and the input size is known.

Every layer in the neural network is created by a Tensor and a mathematical operation that happens in that particular layer. Tensorflow offers a variety of activation functions that can be used for this and all the functions proposed in Section 2.3.1 have been implemented and can be used in each layer. The GUI allows the user to decide the number of hidden layers, the number of neurons in each hidden layer and the activation function in each layer. This is possible in each of the neural network types.

To prevent overfitting in the networks a dropout scheme have been implemented. This is implemented using a Tensorflow component where each output value from a layer has a probability, called the dropout rate, of being set to 0. The dropout rate have been exposed to the user and can be changed before each

run in the GUI. This ensures that the network is not entirely dependant on the value of a few neurons to be able to classify. When each neuron has a chance of being dropped out, the network has to adapt to use more neurons to classify thereby making the system more robust. The dropout scheme is only used during training and not when evaluating the accuracy of the system. This is done to prevent overfitting while training, but when testing the system should perform optimally for classification. The dropout rate is thus set to 0 when testing the system.

A second technique implemented in all the networks is a bias option. When calculating the output of a hidden layer, firstly the previous layers output values are multiplied by the weights leading to the next hidden layer. Then the activation function is applied to the resulting matrix. Implementing a bias is done by creating a matrix of constant values for each layer and adding these values to the resulting matrix just before the activation function is applied. The values of the bias matrix are not dynamically changed during training of the network. This bias option allows the network to learn even more complex models.

Recurrent Neural Network

The main difference between the recurrent network and the standard network is that the recurrent network has a concept of memory. The goal is to create a system that takes into account how the data changes over time. A submarine's environment does not change quickly and this information can be utilized. If a vessel is classified as a ferry at one point, it is reasonable to assume that the same ferry can be heard for a while.

To implement this each layer have to keep some information about the preceding classifications. This is done by creating each layer as a Tensorflow Cell, a structure created in order to build recurrent neural networks. The approach implemented in this project is called a Long short-term memory (LSTM) cell. This cell structure is using additional nodes to keep an internal state as a sort of memory.

The recurrent network implemented in this project is using the sequence of the training data instead of random samples from random ships in a random order. To do this a single recording is divided into several samples and the user specifies how many samples back in time that is relevant to the current classification.

Since the recurrent network is dependant on a sequence of data that is related as input, it can not receive random unrelated samples as input. Therefore an option to use whole files together have been developed, meaning that each 10 second recording is pieced into smaller samples and fed to the network in the same order as the original recording. The user specifies how many samples back in time that is relevant to the current classification and thereby decides the length

of the memory. The total number of samples is less as there is no overlap in these samples.

Convolutional Neural Network

The construction of a CNN has three steps for each layer. The first is to create the Tensor containing the filter from the preceding layer. This filter is otherwise known as the weights and has four dimensions. The filter size is specified by the user and are used as the size of the first two dimensions of the filter matrix. The number of attributes gathered from each patch, called channels, are also set by the user beforehand and this becomes the third and forth dimension in the filter as input and output channels.

The next step is to create a data flow node where the convolution can happen and one for activation. This is where features are extracted from patches of the picture using the filter.

The last piece of each hidden layer is the pooling operation. This is a computation done on the output from the convolution operation. Much like the convolution procedure it is a quadratic filter used stepwise on the convolution data. The difference is that the goal of this operation is to reduce the input stream instead of increasing it. Max pooling and average pooling have been implemented in this project, and is available from the GUI. Max pooling reduces the data where the pooling filter is applied to the maximum value in the patch. The same applies to average pooling, but here the average is kept.

When the network is trained and activated it is crucial that the dimensionality of the following layers match each other to avoid a system crash. The formula used to calculate the resulting number of data points in each dimension is presented in Equation 3.1. The first and second dimension, typically the width and the height of the image, outputted from the first layer is found using this formula. Using this result, the size of the output from the next layer can be calculated. This is done through the entire network and the output size of the last network can be found and used. The formula is also used to calculate how much the data is reduced after the pooling procedure.

$$W_n = (W_{n-1} - F + 2P)/S + 1 \tag{3.1}$$

Where W is the dimension and F is the filter size set by the user. S is the stride or the distance the filter moves and P represents the amount of padding that is done in this dimension, The stride and padding is set as a constant in the configuration file.

At the very end of the network a fully connected standard feed-forward layer is used to reduce the dimensionality of the data before the final classification in

the output layer. The size of this layer can be set in the GUI.

The network have been implemented to receive data in the form of images, namely spectrograms as presented in Figure 3.2 and 3.3. The resolution of these images have to be fixed in order to set the right size to each layer after convolution and pooling as described in Chapter 2. The resolution used in this project was set to 512 data points. This was suitable with a sampling rate of 1024 and sample lengths of 1 second as the resulting spectrogram will be a 32 by 16 pixel image. The system allows for some change as long as the resulting resolution stays the same, e.g. a sampling rate of 512 with a 2 second long sample is also allowed.

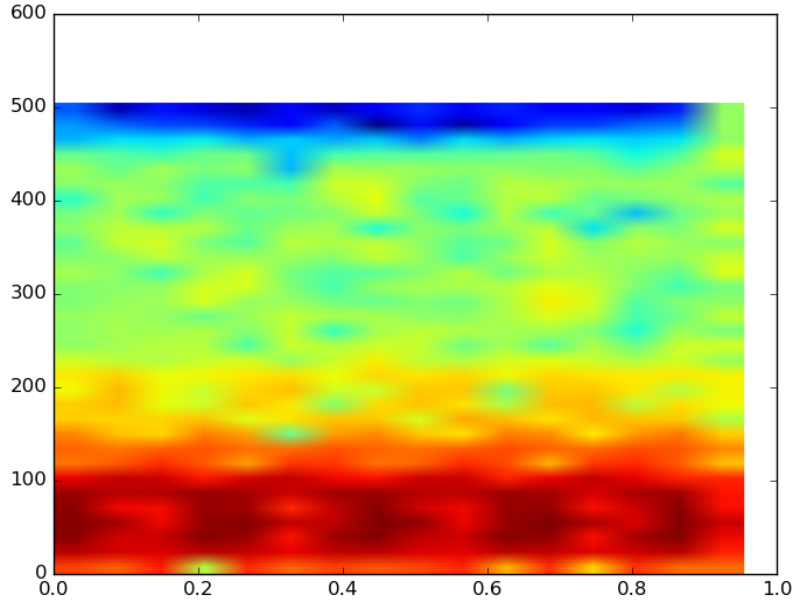

Figure 3.2: A spectrogram image of a sample taken from a ferry. Only this information is passed to the CNN.

3.4.1 Testing the networks

The same procedure have been used to test the classification accuracy of all the different networks. Seven output values are measured after passing through each

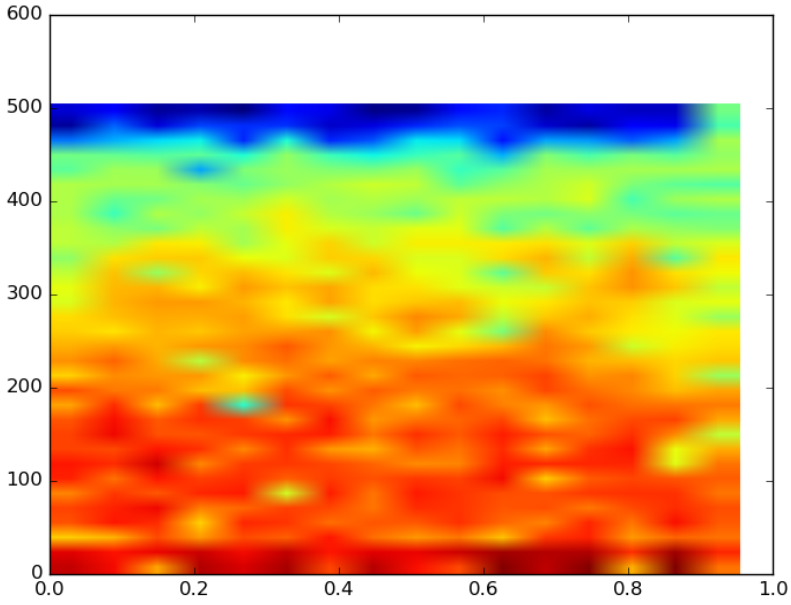

Figure 3.3: A spectrogram image of a sample taken from a destroyer. The destroyer class usually have more activity in the lower frequency bounds than the ferry class.

sample. Each of these seven values correspond to a vessel class and if the correct value has the highest value of the seven, the system has classified the sample correctly. It is highly important to separate test data from training data in order to prevent the system from being trained only to classify data used during training correctly. The ratio of test data contra training data can be set in the configuration file.

In this project a single sound recording of a specific vessel have been divided into several samples. The samples derived from the same recording are unique, but similar to the other samples from the same recording. To use samples very similar to those trained on as test data could lead to a less robust system than expected when confronted with new, less similar data. Three different approaches of separating the entire data set into training and testing data have been implemented. The first is to select random sound recordings and place all samples related to this recording either in the training or the test set. The second approach is to use only the last samples from each recording as test data. When using this approach, a test sample is never surrounded by similar training samples which is important to the quality of the test. The third and last approach is simply to select random samples from random recordings and use these as the test data. This last method have been abandoned in the experimentation in favor of the second, as the data in the test set often have very similar samples in the training set.

3.5 External libraries

Tensorflow

All neural networks have been implemented using Tensorflow, an open-source software library for Machine Intelligence. This is a library where numerical computation is done using data flow graphs. A data flow graph is a concept where the programmer can prepare a system to process data in a predefined way. A directed graph is created where every node is a mathematical procedure and every edge between nodes is the data flow from one node to the next. Using this predefined graph to represent all the computation, Tensorflow is able to organize it in a more efficient way and perform it more efficiently. Another advantage is that Tensorflow offers GPU support, which completes tedious computation in a fraction of the time with a decent graphics card.

Librosa

Librosa is a Python library for audio and music analysis. This is the library responsible for loading the raw audio data from memory and making it possible

to process it as a variable.

SciPy, PyWavelets and Matplotlib

These three are all mathematical Python libraries used to perform the computation needed in order to perform the five different feature extraction techniques. These are all open source libraries.

Chapter 4

Experiments and Results

The goal of this research was to prove that the sonar classification problem could be solved using a neural network, and improved by a preprocessing technique. Since several techniques have been implemented, experimentation is needed in order to find the highest performing solution. The experimentation in this project aims to find an approximation to the optimal parameter configuration in the system. In the first section in this chapter the experimental plan will be presented, followed by the experimental setup and finally the results from the experimentation. A conclusion based on these results will be given in the next chapter.

4.1 Experimental Plan

The experimental plan have been created in order to systematically answer research question 4 as presented in Chapter 1.

RQ4 How will different solutions, or combinations of solutions, affect the overall performance of the system?

There are two types of experimentation that needs to be done in order to answer this. The first is to combine different neural networks and feature extraction techniques. The second is tweaking the relevant parameters in search of the optimal network structure. In order to measure how the change in configuration affects the overall performance of the system, a benchmark, or a starting value, for each combination have to be found. This will be done by running the system with all combinations of FE and NN with a predefined set of default parameter values. These default parameter values are presented in Table 4.1. The default values have been selected as a result of experience with the system accumulated

during development. The aim was to select values that generally worked well in all the possible combinations.

When all the possible combinations here have been tested with the default values and results have been recorded, it is time to start tweaking the parameters. The focus of this experimentation will be to uncover the best possible network configuration, therefore only parameter 10 through 19 are tweaked in the experiments as these will inflict this more than the excluded parameters.

The experimentation will be conducted by changing a single parameter value at the time, first increasing the value than decreasing it. The results will suggest whether the value should be increased or decreased further or stay in between the tested values. This search for optimal values will continue until the positive change converges. The most promising value will be kept and used when experimenting with the next parameter.

The downside of this parameter search is that all the parameters in a network configuration affect each other. The optimal value of a parameter is only found in relation to the others, meaning that the experimentation with the first parameter, the learning rate, will yield an optimal value given the other default values. This value will cascade through the remaining experimentation and all values found will be affected by this. The sequence of the experimentation is important to be able to reproduce the results. The experiments start with the parameter with ID 10 followed in order to 19.

All experiments conducted will be measured by the resulting classification accuracy of the system. The accuracy will be measured two times for each experiment, one for each of the different training-test data separations described in Section 3.4.1. The parameter value yielding the highest score in combination with any feature extraction technique will be kept and used in the remaining experiments. The optimal parameter values found for each network type will be summarized in the final section of this chapter. The final accuracy of each combination will also be recorded when executed using these parameter values.

4.2 Experimental Setup

The question arises, how many times does the system need to be tested with variation in these 10 parameters. The remaining experiments include 2 different training-test separations, 11 FE-NN combinations, 10 parameters, each expected to be changed at least 4 times, not including the initial experiment. This results in a number of 1100 experiments. Some of the parameters are only available when using specific networks, and thus some experiments are abandoned.

The execution of the experiments will be scripted to run sequentially with an altered configuration file for each run, searching for a optimal parameter value. The value leading to the highest accuracy score in any combination will be kept

and used throughout the remaining experiments. All the results from this experimentation will not be presented here, but the best configurations found for each neural network and the results when using this configuration is presented in the next section.

The experimentation environment used was a personal computer where the GPU computations was done using a NVIDIA Gigabyte GTX 770 graphics card. The following Table 4.1 presents the default parameters used in the experiments.

ID	Parameter	Default	Related module
1	Test percentage	0.1	Data loading
2	Sample rate	1024	Data loading
3	Number of samples per file	100	Data loading
4	Length of each sample	1/0.1 sec	Data loading
5	Fast Fourier transform window size	2048	Short-time Fourier transform
6	Enable noise generation	False	Data loading
7	Number of noisy samples per sample	2	Data loading
8	Batch size	10	Neural networks
9	Number of training epochs	100	Neural networks
10	Learning rate	0.001	Neural networks
11	Network depth	2	Neural networks
12	Number of neurons in each layer	256, 128	Standard feed-forward and recurrent neural network
13	Activation function in each layer	Rectified Linear Unit	Neural networks
14	Enable bias	True	Neural networks
15	Dropout keep rate	0.9	Neural networks
16	Time related steps	20	Recurrent neural network
17	Filter size used in between each layer	5, 5	Convolutional neural network
18	Channels extracted from each patch in each layer	32, 64	Convolutional neural network
19	Densely connected layer size	128	Convolutional neural network
20	Padding	2	Convolutional neural network
21	Pooling filter size	2	Convolutional neural network
22	Pooling stride	2	Convolutional neural network

Table 4.1: All configurable parameters and their default value. Parameter 4 have been set to 0.1 for the recurrent network combinations only, as this require shorter related samples.

4.3 Experimental Results

The first results presented here is the classification accuracy of all possible FE-NN combinations when ran with default values. This is recorded in the crossing cells of Table 4.2. Here the end of each file is used as test data for the feed-

forward and convolutional network, and the entire file for the recurrent network. This initial value is used to demonstrate how the system is improving as a better configuration is found.

	Feed-forward	Recurrent	Convolutional
No Feature Extraction	73.76	42.86	X
Short-time Fourier Transform	91.04	59.21	X
Wavelet Transform	74.94	45.71	X
Mel-frequency Cepstral Coefficients	95.29	71.43	X
Spectral Density Estimation	94.94	68.57	X
Spectrogram	X	X	45.14

Table 4.2: Initial test of classification accuracy with default parameters. X means that the combination is incompatible and no experimentation have been done here.

The results from the parameter search is presented in Table 4.3. Here only the 10 parameters experimented with are shown and the parameter ID correspond to the ID given in Table 4.1. The results show that several of the parameters in the final configuration are the same in all the network types. The results show that the system is performing better when using a relative shallow 5 layer network, including the input- and output-layer, than deeper more complex network structures. This observation will be discussed further in the next chapter.

ID	Feed-forward	Recurrent	Convolutional
10	0.0001	0.001	0.0001
11	3	3	3
12	512, 256, 256	512, 256, 256	X
13	Rectified Linear Unit	Rectified Linear Unit	Rectified Linear Unit
14	False	False	False
15	0.9	0.9	0.9
16	X	10	X
17	X	X	5, 5, 5
18	X	X	32, 64, 64
19	X	X	1024

Table 4.3: In this table the best found configuration for each neural network is shown. These configurations are used in combination with the FE techniques and the accuracy of the combinations with this configuration can be reviewed in Table 4.5.

The concluding result in this project is the classification accuracy achieved by the network in combination with feature extraction, when using the found optimal configuration. These results are presented in two tables where the first, Table 4.4 contains the accuracy found when the testing is done on whole files. The second and final summary of the results gathered in this project is presented in Table 4.5. The classification accuracy of the combinations running with the found optimal configuration and the preferred training-test data separation, namely using samples gathered from the end of the recordings as test data.

	Feed-forward	Recurrent	Convolutional
No Feature Extraction	38.71	42.86	X
Short-time Fourier Transform	45.29	54.29	X
Wavelet Transform	34.14	48.57	X
Mel-frequency Cepstral Coefficients	53.43	77.14	X
Spectral Density Estimation	48.86	42.86	X
Spectrogram	X	X	46.29

Table 4.4: The results when the system is ran with optimal configuration. Entire files are used as test data.

	Feed-forward	Recurrent	Convolutional
No Feature Extraction	69.22	42.86	X
Short-time Fourier Transform	90.24	54.29	X
Wavelet Transform	65.65	48.57	X
Mel-frequency Cepstral Coefficients	97.29	77.14	X
Spectral Density Estimation	94.82	42.86	X
Spectrogram	X	X	85.17

Table 4.5: The classification accuracy of each combination ran with the best found configuration for each network type. The highest scoring accuracy found was with the use of MFCC and a standard feed-forward network at 97.29%. The end of each file is used as test data.

The configuration found and used here achieved 97.29 % classification accuracy when combining a standard feed forward network and the MFCC feature extraction technique. This is exactly 2 % better than the same combination used with default parameters, showing the importance of the parameter search. This configuration did not outperform the default configuration when used in all FE-NN combinations, but achieved the highest classification accuracy in a single combination.

The recurrent neural network are only tested with the whole files as test data, as the sequence of the samples are used for training. Testing on random samples or only the end of the files would result in a system trained for sequential classification and tested without utilizing this information. The other two network types have also been tested with this separation in order to compare the results from the recurrent network.

Chapter 5

Evaluation and Conclusion

This chapter presents an evaluation of the research and the system developed. Then the conclusions drawn from the results presented in Chapter 4 are discussed in Section 5.2. This is followed by a summary of the contributions given in this thesis in Section 5.3. Lastly the recommendations for how to proceed with the further work are given in Section 5.4.

5.1 Evaluation

In Chapter 1, the goal of this project was defined as follows:

Goal Develop a system where a signal preprocessor and a neural network can work together to classify marine vessels based on their acoustic signature.

To achieve this goal a system was developed and is presented in detail in Chapter 3. In this system a number of different techniques were implemented, both related to neural networks and to feature extraction.

The preprocessing problem were more often than not solved by using a time-frequency transformation to transform the data into the frequency-domain. This yielded features about the frequency bands found in the data, and has proven to be a powerful way to extract informative features from sound data. To confirm this, all the networks were trained using raw unprocessed data, and the networks ability to classify correctly were tested. It was found that the classification accuracy improved greatly when a preprocessing technique was introduced. This was the general case for all the implemented techniques, except in some cases where the wavelet transformation was used. This proved to be the least efficient technique tried in this project. Using the Mel-frequency Cepstral Coefficients

technique with the best found network configuration improved the classification accuracy by 28.07% in comparison with no feature extraction technique.

The solutions found to the classification problem were more diverse. They differed in the complexity of the data to be classified and the computational cost of training. The networks have been implemented to handle the output data from the preprocessing techniques, without knowledge of how the data have been processed. This is a great advantage with neural networks, it is possible to feed them unstructured data from the feature extractor, and the network itself will be able to recognize the identifying patterns.

To combine the two components have shown to be less problematic than anticipated as the neural networks is found to be very flexible when it comes to what kind of data they can process and recognize features in. The project have been evaluated through experimentation presented in Chapter 4. The goal was to find combinations suitable for a task, and the experimentation has shown that the results vary widely between different combinations. The variation in classification rate suggests that some combinations are better suited than others. The combination yielding the highest classification accuracy was the combination of a MFCC preprocessor and a standard feed-forward neural network with three hidden layers. The final classification accuracy was 97.29%.

5.2 Discussion

As a result of the research done in this project a system have been developed, able to differentiate between 7 different vessel-types based only on the sound they emit in water. The system have been used to systematically search for the best combinations between feature extraction technique and neural network configuration. This system was made as a prototype, in order to test the feasibility of the techniques applied on sonar data. The purpose have been tested to the extent where it is possible to identify techniques more suitable than others.

The combination found with the highest accuracy are not the most complex one, using a standard feed-forward network with only three hidden layers. This could be due to the complexity of the classification task. The complexity needed in a network is typically lowered when preprocessing of the data is performed. The aim of the preprocessing is to extract only the informative features from a larger data set, and remove redundant information leaving the network with less data to process.

The experimentation also show a great difference in accuracy when using different training-test data separation schemes. When using entire files as test samples, the accuracy drops significantly in all tests. This could indicate that the data set is too small, and the difference in each recording is too large. When given access to more raw sonar data the system would be able to use this for

training and become more robust and capable of classification of further classes.

The feature extraction techniques implemented have been found to be state-of-the-art solutions to environmental sound processing. The techniques were not specialized to handle the sonar data recorded in deep water, but selected for their ability to extract features from sound recorded in a variety of different scenarios. The results presented in Chapter 3 points to the conclusion that the techniques implemented are suitable to use with the sonar data as the results are improved by 28.07% when using the best network configuration found.

The MFCC feature extraction technique was in the structured literature review discovered mainly for it's applications in speech recognition systems. It was suitable for this task as it was developed as a technique specializing in extraction of features from sound rich in modulation, like speech. The technique was kept after reviewing Uzkent et al. [2012] and Cowling and Sitte [2003] where the technique were successfully used to analyze environmental sounds. The data used in this project is similar to environmental sounds as continuous noise is present, but the success of the MFCC method suggests that the identifying characteristic of each vessel type lies in the modulation of the sound.

Of the three neural network types used in this project, the feed-forward network were most commonly recommended in the literature review. This could be because many of the studies found were studies not primarily in the field of AI, but in the field of audio analysis. This network implementation is the least advanced of the three, but it still achieves the highest scores in the experimentation. The recurrent network achieved a classification accuracy of 77.14% at it's highest configuration. When files are divided into smaller sequenced samples this is the highest score achieved, suggesting the potential of the recurrent neural network. Interestingly this is also in combination with the MFCC method, once again indicating that this is the most suitable feature extraction technique for sonar data.

The most complex neural network model implemented was the convolutional neural network. To utilize this kind of network the data fed into it have to have a structure where there is a relation between data points close to each other. A structure where this is typically the case is images. Using a spectrogram as the input of this network showed successful with an achieved classification accuracy of 85.17%. In an on board scenario, the spectrogram resolution could be set much higher as continuous data is recorded and can be processed into a spectrogram with the desired resolution in time. A spectrogram of higher resolution could require a more complex convolutional network structure in order to be interpreted more accurately.

The system developed was intended to assist the sonar operator by searching every direction at once and quickly suggest the class of the surrounding vessels. This has proven to be feasible as the system is able to suggest a class, given some

input data, with a very high accuracy in a fraction of a second. The time used to train the system is dependant of the size of the training data set used, but is only required once before the system becomes capable of classification. The operative system on board could then make classifications continuously in all directions.

5.3 Contributions

Based on our research goals and questions presented in Chapter 1, the contributions made in this project can summarized in the following way:

The main contribution is the system developed, able to classify 7 different maritime vessels based on their emitted sound with an accuracy of 97.29%. This system is also able to test a variety of configurations for research purposes. Three different neural network types have been implemented along with five signal-processing feature extraction techniques. Dropout, bias, noise and data loading have are also made available in the system.

In order to develop a system specialized in underwater environmental sound both the fields of signal processing and artificial intelligence is explored and the background information collected is summarized in Chapter 2. The background information have been gathered through a structured literature review performed by the author of this thesis, Gimse [2016]. This research have been the second contribution done in this project.

5.4 Future Work

This section propose possible improvements to the system and the project. The system have been built mainly for research purposes, but the results and analysis can be used in order to create a live system on board a submarine. As the project has a modular architecture, it is possible to develop new components to handle this with the existing system and even develop new feature extraction techniques and networks that can be used with the same system.

5.4.1 Data size

Training with neural networks require a large data set to train with. The data set used here were only a small number of recordings divided into shorter samples. The number of samples were initially not large enough, but this was solved by allowing each sample to have some overlap to another sample. The system would greatly improve if this was not necessary as many similar samples can lead to overfitting in the model.

5.4.2 Recurrent network

The recurrent neural network is designed to perform best when the task requires memory. To arrange this classification task in such a way, the recordings were divided and used in sequence. This made every sample the recurrent network received very small and classification thus became more inaccurate. This may be the reason the recurrent neural network did not perform as good as the other networks. In a more realistic continuous data scenario this would not have been a problem. This improvement is recommended for future work.

5.4.3 Speed and Direction

The system implemented does not have a concept of the relative speed and direction of the vessel it is trying to classify. This is a very important part of submarine navigation and with some further work, and training data where this is labeled, it would be possible to suggest this information as well without any additional data than the sonar data recorded continuously on board.

Appendix A

Acronyms

AI Artificial Intelligence

NN Neural Network

FE Feature Extraction

STFT Short-time Fourier Transformation

MFCC Mel-frequency Cepstral Coefficients

LSTM Long Short-term Memory

Bibliography

Britz, D. (2015). Understanding convolutional neural networks for nlp. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. Accessed: 2016-05-02.

Cowling, M. and Sitte, R. (2003). Comparison of techniques for environmental sound recognition. *Pattern Recognition Letters*, 24(15):2895–2907.

Gimse, H. (2016). Classification of marine vessels using sonar data and a neural network, a structured literature review. http://folk.ntnu.no/haakongi/Forprosjekt.pdf. Accessed: 2017-03-26.

National Instruments (2015). Acquiring an analog signal: Bandwidth, nyquist sampling theorem, and aliasing. http://www.ni.com/white-paper/2709/en/. Accessed: 2016-05-01.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

The International Institute of Marine Surveying (2015). An introduction to propeller cavitation. http://www.iims.org.uk/introduction-propeller-cavitation/. Accessed: 2016-05-01.

The National Physical Laboratory (2005). Speed of sound in sea-water. http://resource.npl.co.uk/acoustics/techguides/soundseawater/content.html. Accessed: 2016-03-25.

Uzkent, B., Barkana, B. D., and Cevikalp, H. (2012). Non-speech environmental sound classification using svms with a new set of features. *International Journal of Innovative Computing, Information and Control*, 8(5 B):3511–3524.

Welch, P. (1967). The use of the fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15:70–73.