Evaluation of a deep neural network for acoustic classification using simulated echo sounder data

by
Taran Fjell Naterstad

Master of Science Thesis in
Applied and Computational Mathematics

Department of Mathematics
University of Bergen

June 2020

Abstract

An important part of fisheries acoustics is the classification of fish species. Sound waves are transmitted through water to detect fish species, and the echoes returning from the fish are categorized to be used for fish abundance estimates. These estimates are import for fishery management. Recently, it has been shown that a deep learning model performs well on the task of classifying acoustic data. However, these models are often criticized for being "black boxes" and hard to interpret. We have created a pipeline to test a neural network model, in order to shed light on what features of the data impact the predictions of the model. In this pipeline, simulated data is utilized, created by a model that emulates the performance of a multi-frequency echo sounder. The simulated data enables the possibility of adjusting one feature of the data at a time. We have concentrated on two features: the relative frequency response, an energetic characteristic of the data, and the shape of the fish schools. A neural network is trained to recognize two types of fish schools, dissimilar only in shape and relative frequency response. The network is then tested on data where either shape or relative frequency is changed, to evaluate the importance of each feature. From these tests we conclude that the relative frequency response affects the model's performance more than shape.

Acknowledgments

I would like to thank my supervisors, Jarle Berntsen and Nils Olav Handegard. Your guidance, encouragement and critique has been crucial for the completion of my master thesis. Thanks also to Arne Johannes Holmin, for your knowledge and help with the simulation model. A big thank you to everyone else that I have met at the Institute of Marine Research, you all made me feel very welcome. A special thanks to all of my friends and family, for your unwavering support.

Contents

List of Figures

List of Tables

List of Symbols

α	Intensity absorption coefficient
η	Orientation factor of an acoustic target
λ	Wavelength
σ_{bs}	Backscattering cross section
B_l	Beam pattern of an acoustic target
B_T	Beam pattern of a transducer
c	Wave speed
f	Frequency
I	Intensity
J_1	Bessel function of the first kind
k	Wavenumber
L	Size of an acoustic target
l	Length of swimbladder
P	Power
r	Radius of a sphere
$r(f)$	Relative frequency response
s_v	Volume backscattering coefficient

Abbreviations

FN	False negative.
FN	False positive.
KL	Kullback-Leibner (divergence).
PR	Precision-recall (curve).
SDG	Stochastic gradient descent.
TN	True negative.
TP	True positive.
TS	Target strength.

Chapter 1

Introduction

1.1 Introduction

Acoustic trawl surveys locate and estimate fish biomass for stock assessment and ecosystem studies (Simmonds and MacLennan, 2005). Measurements are performed using echo sounders, a special case of sonar where the acoustic beam is directed vertically in the water column. Echo sounders transmit pulses of sound at either a single or multiple frequencies to remotely detect objects in water. These sound waves propagate through the water until they reflect of a target or the seabed. The reflected sound, also referred to as backscattered sound, is detected by the echo sounder. The echo sounder displays the backscattered sound as echograms, two-dimensional arrays that show the position and morphology of the targets, and give information about the amplitude of the backscattered sound. To estimate the abundance of a species, the different marks of the echogram have to be categorized. In order for the estimate to be precise, the classification has to be done correctly. Classification can not be done based only on the echogram, since the echo amplitude of a target is dependent on several features such as the transmitting frequency of the echo sounder, target morphology, the tilt of the target (whether the fish is swimming up or down), and the presence or absence of a swimbladder. Typically, this is done manually using knowledge of the local fish populations, and aided by trawl samples (Simmonds and MacLennan, 2005).

Manual target classification is time consuming and prone to human bias, and a range of classification methods have been proposed to automate the process and reduce subjectivity. Weill et al. (1993) used principal component analysis and linear discriminant analysis for classification of fish, while Haralabous and Georgakarakos (1996) trained an artificial neural network. Both of these studies used features gathered from the fish aggregations, such as energetic and morphological characteristics.

Another important feature used in categorization of species is the frequency dependence of targets, called the relative frequency response, defined as the ratio of the backscattered energy at frequency f to an reference frequency f_0, usually 38 kHz; $r(f) \equiv s_v(f)/s_v(38\text{kHz})$. This has been used for classification of deep-water orange roughy (Kloser et al., 2002), and classification of herring, mackerel, and capelin (Korneliussen and Ona, 2002; Korneliussen and Ona, 2003; Fernandes et al., 2006; Korneliussen et al., 2009). Other methods, such as random forest and k-means clustering, have also been used for categorization (Fallon et al., 2016; Gastauer et al., 2017).

Another possible solution for target classification is the use of deep convolutional networks. These network are composed of multiple processing layers that learns representations of data, i.e., they do not use engineered features as the methods mentioned above, but rather learn the features needed for classification from the raw data (Lecun et al., 2015). Brautaset et al. (2020) showed that the U-Net model can be used for categorizing acoustic multi-frequency echo sounder observations. The U-Net model was first proposed for the purpose of segmenting blood cells (Ronneberger et al., 2015). This fully convolutional network consists of an encoder and a decoder. The encoder maps the input to a low-resultion representation, and the decoder is a mapping from the low-resolution representation to a pixel-wise representation. However, as is with all neural networks, the model is hard to interpret and does not provide information about feature importance. It is therefore less transparent than more conventional methods, such as random forest, where hand-crafted features are used (Brautaset et al., 2020).

The objective of this thesis is to expand on the works of Brautaset et al. (2020), by trying to identify the significance of two features of the fish schools from the echo sounder observations; relative frequency response and shape. The goal is to determine if one is of more significance than the other, or if the features are equally important when the network classifies fish schools. To achieve this, we need data where we can adjust one feature at a time. Therefore, a simulation model that emulates acoustic data from a multi frequency echo sounder is used to create a data set. The data will include two types of fish schools that differ only in shape and relative frequency response. We will train a neural network to classify the acoustic data. Lastly, we will observe the model's performance on data where the fish schools have been changed in either shape or relative frequency response.

1.2 Chapter overview

Chapter 1 – Introduction This chapter.

Chapter 2 – Background This chapter gives the reader basic knowledge of the concepts used in this thesis. A short introduction to fisheries acoustic is given in Section 2.1, then some key concepts within neural networks are explained in Section 2.2.

Chapter 3 – Materials and methods The simulation model used to create a data set is explained in Section 3.1. In Section 3.2 the data set used to train and test the neural network model is explained, along with the training scheme and architecture of the model. How the model performance is tested is explained in Section 3.3.

Chapter 4 – Results The performance of the model is described. First, the performance of the model on data similar to data it has been trained on is discussed in Section 4.1. Then, the performance of the model on perturbed data is described in Section 4.2.

Chapter 5 – Discussion The results and their implications are discussed.

Chapter 2

Background

This chapter is meant as a short introduction to terms within fisheries acoustics and machine learning. Section 2.1 introduce concepts within fisheries acoustics. Section 2.2 introduces terms within machine learning, specifically those connected to the field of neural networks.

2.1 Fisheries acoustics

2.1.1 Echo sounders

Acoustic surveys are done using echo sounders to detect or observe remote objects in the sea. Sonar is the general term for all devices capable of remote detection in water, while an echo sounder is a special implementation of sonar where the acoustic beam is directed vertically downwards.

Figure 2.1 shows the different parts of the echo sounder. The transmitter creates a burst of energy for a specified frequency. The transducer converts the energy into a sound wave that propagates through the water column. Targets, such as fish, plankton or the seabed, reflect and scatter the pulse of sound. This backscattered sound is received by the transducer again, which converts it to electrical energy. The signal is then amplified and displayed on an echogram. The depth of the target is calculated from the travel time of the pulse from the transducer to the target and back again.

2.1.2 Echograms

The echogram consists of vertical lines corresponding to each transmission. If the transducer is fixed, then the echogram will be a time-series of a specific volume. To detect fish, the transducer is moving at constant speed in one direction, and the

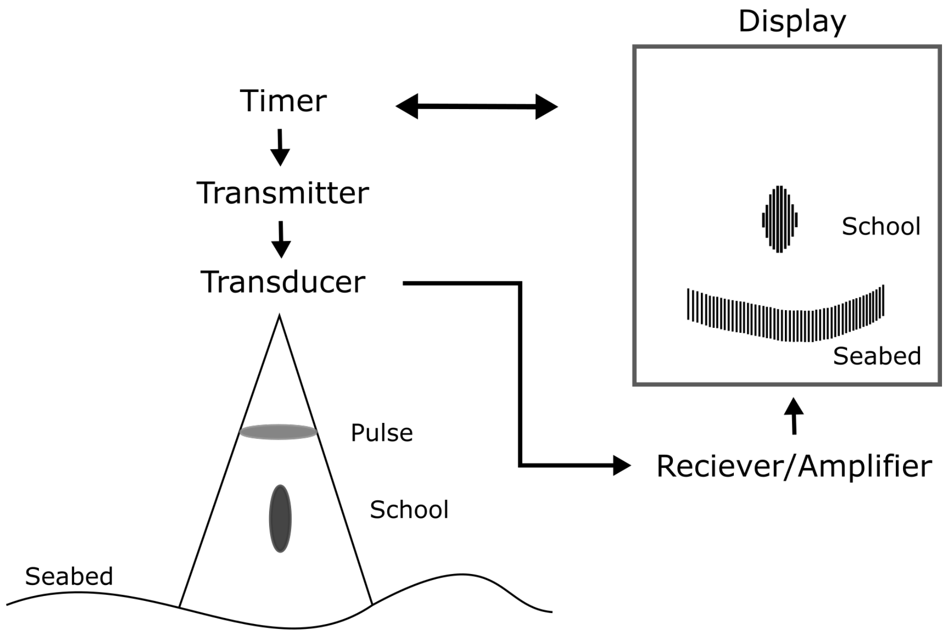


Figure 2.1: An illustration of how an echo sounder works. A sound wave is propagated through the water, and reflected off of targets such as fish. The reflected sound is then used to create an echogram, a vertical cross-section of the water column.

echogram is a cross-section of the water column. Each of these vertical lines show how the acoustic reflectivity varies for each transmission. If a target is detected, it will show up as a mark on the echogram. With support from trawl samples and knowledge of species composition, the mark can be linked to a specific species or group of species (Simmonds and MacLennan, 2005).

2.1.3 Acoustic propagation

Beam spreading

Sound waves spread as they propagate through water. This means that the intensity, i.e., power transmitted through a unit area, is reduced as the wave gets further away from the transducer.

In a lossless medium and with sound waves coming from a point source, the power of the wave P will radiate in all directions. As there is no power lost in the medium, the power P is constant,

$$P = \iint_A I \cdot dA, \tag{2.1}$$

where I is the intensity as a function of distance, and dA is the differential element of a closed surface A that contains the sound source. For the point source mentioned above, this area will be the surface of a sphere. If the intensity is uniform, Equation (2.1) will become

$$P = I_r \cdot 4\pi r^2,$$

where I_r is the intensity at distance r, the radius of the sphere. As this must hold true for any distance r, we can show that intensity follows the inverse-square law. If I_0 is the intensity at a distance of 1 meter from the point source, and I_r is the intensity at distance r, and P is constant at any range r, then

$$I_0 \cdot 4\pi 1^2 = I_r \cdot 4\pi r^2,$$

which leads to

$$I_r = I_0/r^2,$$

i.e., intensity is proportional to the range squared.

If the acoustic waves are spreading within a given angle Ω, the intensity of the waves will follow the same law, given that we are still in the same medium. Due to the fact that the surface area of the section of the sphere within Ω increases proportional to the radius squared in the same way as before.

Absorption

As a sound wave makes its way through the water, some of its acoustic energy is lost due to absorption; conversion from acoustic energy to heat
(Simmonds and MacLennan, 2005).

The loss can be written as

$$\frac{\mathrm{d}I}{I} = -2\delta \mathrm{d}x, \tag{2.2}$$

where $\mathrm{d}I/I$ is the fractional infinitesimal change in intensity, δ is called the pressure attenuation coefficient, and $\mathrm{d}x$ is the infinitesimal distance traveled (Kinsler et al., 2000). If we integrate Equation (2.2) from distance x_o to x, we will get the expression

$$\ln I(x) - \ln I_0 = -2\delta(x - x_0),$$

where $I_0 = I(0)$. Exponentiating both sides results in

$$I(x) = I_0 e^{-2\delta(x-x_0)}. \tag{2.3}$$

A more common way to write Equation (2.3) is obtained by first taking the logarithm of base 10 of both sides of the equation,

$$\begin{aligned}\log_{10} I &= \log_{10} I_0 - 2\delta(x - x_0)\log_{10}(e) \\ &\approx \log_{10} I_0 - 0.869\delta(x - x_0) \qquad (2.4)\end{aligned}$$

where in Equation (2.4) we have used that $2\log_{10}(e) \approx 0.869$. We then raise to the power of 10 to get the final equation

$$I(x) = I_0 10^{-\alpha(x-x_0)/10}, \tag{2.5}$$

where $\alpha = 8.69\delta$ is the (intensity) absorption coefficient, expressed as energy loss in dB per unit distance. Often, x_0 is negligible compared to x and Equation (2.5) is typically written as

$$I(x) = I_0 10^{-\alpha x/10}.$$

Absorption is primarily dependent on frequency, with higher absorption for higher frequency, but water salinity and temperature are also contributing factors (Simmonds and MacLennan, 2005).

2.1.4 Acoustic scattering

As mentioned in Section 2.1.1, targets scatter or reflect the sound wave transmitted from the transducer of an echo sounder. The part of a scattered sound wave that is reflected back to the echo sounder is referred to as backscattered sound. This provides the sonar echo in the case where the transducer is used for both transmission and reception. There are different types of scattering that can occur when a sound wave encounters a target.

Whenever a target is small compared to the wavelength λ of the incident sound wave, the whole target will be subject to the same sound pressure. The pressure oscillations of the incident wave will make the target contract and expand in response, turning the target into a point source of the scattered waves. These waves will then spread spherically in all directions. With small targets, it is mostly the volume of the target that determines the scattering. If L is the size of the target, then the scattered energy is proportional to $(L/\lambda)^4$ whenever $L \ll \lambda$. This is called the Rayleigh scattering law.

If instead the target is much larger than the wavelength, $L \gg \lambda$, the surface of the target will reflect the incident wave rather than the volume. With a smooth, plane surface, the incident wave will simply be reflected, following the rule of equal angles for incidence and reflection. The scattering is then referred to as specular scattering. More likely, the target is spherical, and the scattered energy of target will approximately increase as the square of the radius of the sphere. Whenever this happens, we call the scattering geometric.

In the case when target size and wavelength are similar, $L \approx \lambda$, the scattering depends on both the geometric structure and the material properties of the target. The strength of the scattering can change rapidly with frequency due to resonances that can occur.

In short, the scattering from small targets increase with frequency, while for large targets the scattering is less dependent on the frequency. If size and wavelength are similar, then resonances will occur, which makes it difficult to predict the scattering from the target.

2.1.5 Acoustic properties of fish

The strength of backscattered sound from a target is described as backscattering cross section or target strength (TS).

Backscattering cross section

Backscattering cross section at a distance r from the sound source (σ_{bs} in units m^2) is defined as

$$\sigma_{bs} = r^2 \frac{I_b}{I_i},$$

where I_b is the sound intensity reflected or backscattered from the target, and I_i is the intensity of the incident pulse measured at an arbitrary distance, usually 1 m (Simmonds and MacLennan, 2005).

Target strength

Target strength is another way to express echo backscattered from the target, defined as the logarithmic transformation of the backscattering cross section,

$$\text{TS} = 10\log_{10}(\sigma_{bs}).$$

Measuring TS in decibels keeps its range relatively short. It is usually between -60 dB and -20 dB, even though the size of targets can differ greatly, from plankton to whales (Simmonds and MacLennan, 2005).

From target strength experiments, a relationship between target strength and length of the target has been found to be reasonable and convenient (Simmonds and MacLennan, 2005). The relationship can be expressed as

$$\text{TS} = m\log_{10} L + b,$$

where L is the target length from the front of the head to the tip of the tail, and m and b are constants for a given species. Both m and b can be estimated by linear regression of target strength on $\log_{10} L$, given data from different groups of fish with a range of mean lengths (Simmonds and MacLennan, 2005). The coefficient m tends to be between 18 and 30, and is often close to 20. This has resulted in a standard formula of the form

$$\text{TS} = 20\log_{10} L + b_{20}, \tag{2.6}$$

where b_{20} is called the reduced target strength. This can be estimated as the mean of (TS-20 $\log_{10} L$) (Simmonds and MacLennan, 2005).

Volume backscattering coefficient

When individual targets are small and clustered, their echoes combine to form a received signal that is continuous with varying amplitude. One cannot resolve individual targets anymore, but the echo intensity is still a measure of the biomass in the water column. The basic acoustic measurement is the volume backscattering coefficient, s_v, formally defined as

$$s_v = \sum \sigma_{bs} \ / \ V_0,$$

where the sum is taken over all the discrete targets contributing to echoes from V_0, the sampled volume (Simmonds and MacLennan, 2005).

Relative frequency response

The relative frequency response is an acoustic feature used to group acoustic backscatter into acoustic categories. It has been successfully applied to multi-frequency data to distinguish broad acoustic categories (Korneliussen and Ona, 2003). The relative frequency response for a frequency f is defined as $r(f) = s_v(f)/s_v(f_0)$, where $f_0 = 38$kHz. In the case of a single target, $r(f)$ will simplify to $r(f) = \sigma_{bs,f}/\sigma_{bs,f_0}$. Figure 2.2 shows the expected relative frequency response for a few target categories. The solid line is the backscatter from fluid-like objects, objects that don't differ much from seawater when it comes to sound speed and density. As shown in the figure, backscatter from these objects fluctuate in the region between low-frequency scattering regions (Rayleigh scattering region) and the high-frequency regions (geometric scattering region). Rayleigh scattering and geometric scattering are explained in Section 2.1.4. Targets that are gas-filled, e.g., fish with swim bladders, produce a resonant scattering at a frequency which is dependent on the size and depth of the gas inclusion. The line with long dashes, the backscatter from elastic-shelled zooplankton, has a smooth transition from the Rayleigh scattering region to the geometric scattering region. Scattering classes for the frequency range from 18 kHz to 200 kHz are marked on the figure in the region where they are expected. Some simplifications have been done, i.e., the three curves will not follow each in the low-frequency region as shown, due to difference in slope. Furthermore, there will be differences within each target class, i.e., the rate of increase, height and width of the resonance peak for gas-filled targets (Korneliussen and Ona, 2003).

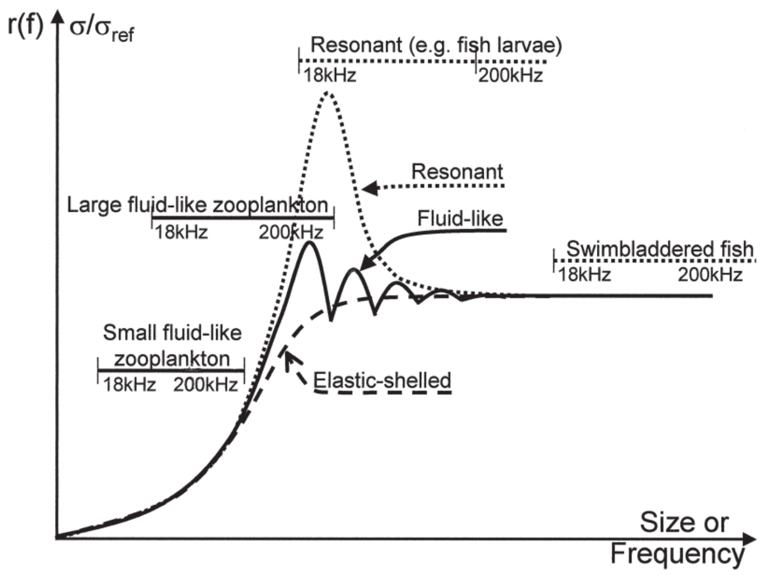


Figure 2.2: Expected relative frequency response, $r(f)$, for a few target categories. Reprinted with permission from R. J. Korneliussen and E. Ona, "Synthetic echograms generated from the relative frequency response", *ICES Journal of Marine Science* 60.3, 2003, by permission of Oxford University Press.

2.2 Neural networks

Neural networks are models that, given an input vector $\mathbf{x}$ and an output vector $\mathbf{y}$, try to approximate some function $\mathbf{y} = f^*(\mathbf{x})$. The network is a mapping $\hat{\mathbf{y}} = f(\mathbf{x}; \boldsymbol{\theta})$, where the set of parameters $\boldsymbol{\theta}$ give the best function approximation. If the flow of information goes from $\mathbf{x}$ to $\hat{\mathbf{y}}$, and no information from the output $\hat{\mathbf{y}}$ is fed back to the network it is said to be *feedforward*. The word network comes from the fact that these models are a combination of several functions, $f(\mathbf{x}) = f^n(f^{n-1}(...(f^1(\mathbf{x}))))$. The functions are termed *layers*. Here, $f^1(x)$ is the first layer of the network, $f^2(f^1(\mathbf{x}))$ the second, and so on. The vector $\mathbf{x}$ is called the input layer, f^n is called the *output layer*, while the other layers are termed *hidden layers* as their output is generally not seen (Goodfellow et al., 2016).

2.2.1 A fully connected network

We will describe a small network consisting of an input vector $\mathbf{x}$, a hidden layer $\mathbf{h}$ and an output layer $\hat{\mathbf{y}}$, such as the network seen in Figure 2.3. Every component of the hidden layer has two parts; a weighted summation of the input and an activation function g. If the activation function is the identity function $g(x) = x$, the model will become a linear model. To avoid this, the activation function is used

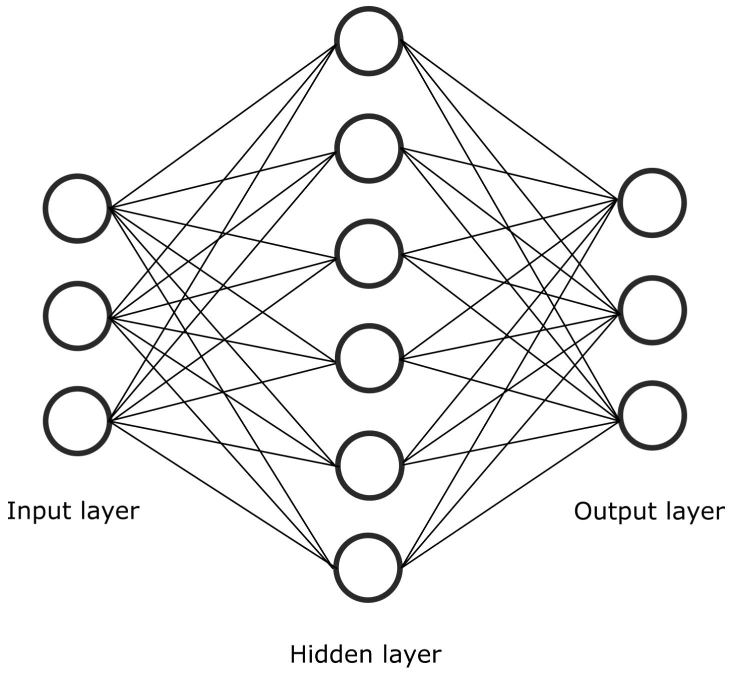


Figure 2.3: A fully connected neural network consisting of an input layer, a hidden layer, and an output layer. A fully connected layer is dependent on all the elements of their previous layer.

to introduce non-linearity to the model in order to achieve greater computational flexibility (MacKay, 2003).

If k spans the dimension of the input vector $\mathbf{x}$ and j denotes the unit of the hidden layer, the equations of the hidden layer can be expressed as

$$a_j^{(1)} = \sum_k w_{kj}^{(1)} x_k + b_j^{(1)}; \quad h_j = g^{(1)}(a_j^{(1)}), \tag{2.7}$$

where the superscript denotes the layer of the network and b_j is the bias of the unit j. In the same way, the equations of the output layer of the network can be expressed as

$$a_i^{(2)} = \sum_j w_{ji}^{(2)} h_j + b_i^{(2)}; \quad \hat{y}_i = g^{(2)}(a_i^{(2)}), \tag{2.8}$$

where i spans the number of output units. The weights $\mathbf{w}$ and biases $\mathbf{b}$ are often expressed as the parameter vector $\boldsymbol{\theta}$. Finding the $\boldsymbol{\theta}$ that makes the model fit the input data well is called *learning* and the input data is called *training data*. For our small network, Equations (2.7) and (2.8) describe a *forward pass*; the mapping from $\mathbf{x}$ to $\hat{\mathbf{y}}$. The layers described in Equations (2.7) and (2.8) are also called *fully connected*, as they are dependent on all the elements of their previous layer, illustrated in Figure 2.3.

Learning

Training a network means adjusting its weights in such a way that the error between the desired output and the actual output from the model is reduced (Hinton, 1992). This error is often referred to as the cost function. To achieve this, it is necessary to know how the error changes when each weight is perturbed. In other words, the neural network must calculate the error derivative of the weights (Hinton, 1992). The most popular method for this is the backpropagation method.

Let the error function be the square sum error,

$$E = \frac{1}{2} \sum_i (\hat{y}_i - y_i)^2,$$

where $\hat{y}_i$ is the ith unit of the output layer described in Equation (2.8), and y_i is the desired or true value of the unit. As discussed above, letting the bias $\mathbf{b}$ correspond to the weight of an input $x_0 = 1$, we can express the bias and weights as a vector $\boldsymbol{\theta}$. Using Equations (2.7) and (2.8), we can calculate the error derivative of the weights for the different layers using the chain rule of calculus. We start by finding the expression for $\frac{\partial E}{\partial \theta_{ji}^{(2)}}$,

$$\frac{\partial E}{\partial \theta_{ji}^{(2)}} = \frac{\partial E}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial \theta_{ji}^{(2)}}. \tag{2.9}$$

Typically, gradient descent is used to minimize the error E. In its simplest form, gradient descent changes the weights by an amount proportional to the accumulated $\partial E/\partial\theta_{ji}^{(2)}$,

$$\theta_{ji}^{(2)} \leftarrow \theta_{ji}^{(2)} - \epsilon\partial E/\partial\theta_{ji}^{(2)},$$

where ϵ is called *learning rate*.

To update the weights of the first layer, the chain rule is applied again. Instead of taking the derivative of $a_i^{(2)}$ with respect to the weights in Equation (2.9), the derivative of $a_i^{(2)}$ with respect to h_j is calculated. That way the chain rule can be applied to find $\partial E/\partial\theta_{kj}^{(1)}$. In addition, all the connections from the output layer to the hidden unit h_j is summed, as they all contribute to the derivative $\partial E/\partial\theta_{kj}^{(1)}$,

$$\frac{\partial E}{\partial\theta_{kj}^{(1)}} = \sum_i \frac{\partial E}{\partial\hat{y}_i}\frac{\partial\hat{y}_i}{\partial a_i^{(2)}}\frac{\partial a_i^{(2)}}{\partial h_j}\frac{\partial h_j}{\partial a_j^{(1)}}\frac{\partial a_j^{(1)}}{\partial\theta_{kj}^{(1)}}.$$

Continuing to apply the chain rule in this manner, gradient descent can be used to minimize the cost function in all neural networks, even though they are much deeper than our example here.

2.2.2 Convolutional neural networks

When working with image data, or other 2-dimensional arrays, the number of parameters needed in a fully connected network will quickly become very large. Every element of the array, or every pixel, would have a weight associated to it. For these data sets, *convolution neural networks* have been shown to have good results (Krizhevsky et al., 2012). In these networks, the neurons of a layer does not depend on every element of the previous layer, but a subset of these, called the *receptive field*. This reduces the number of weights that needs to be learned during training.

Convolutions is a type of spatial filtering; it replaces each pixel with a function of the value of the pixel and its neighbors (Gonzalez and Woods, 2018). These filters can be handcrafted to detect certain features, such as lines or corners. In a neural network, the goal is to use backpropagation in order to find the features that represent the data the best. In other words, letting the network learn the features of the data, instead of handcrafting them. By stacking convolutional layers, increasingly abstract features of the data can be recognized.

The equations of a convolutional layer

Keeping the notation from before, let $h_{x,y}$ be the image feature value from a previous layer. The linear spatial filtering of the point (x, y) in the input, by a kernel $\mathbf{w}$ of size $m \times n$, can be expressed as

$$w \star h_{x,y} = \sum_{l=0}^{m} \sum_{k=0}^{n} w_{l,k} h_{x+l,y+k}, \tag{2.10}$$

also referred to as the *spatial correlation* of the image. *Spatial convolution* consist of the same computation, but with the kernel rotated 180 degrees (Gonzalez and Woods, 2018). In many neural network libraries, the spatial correlation expressed in Equation (2.10) is implemented, but referred to as a convolution (Goodfellow et al., 2016). We will follow this convention here. Let the kernel w be 3×3, then for a specific element of a 2-dimensional array at position (x, y), Equation (2.10) performs the sum of product of the form

$$\begin{aligned} w \star h_{x,y} &= w_{0,0} h_{x,y} + \cdots + w_{3,3} h_{x+3,y+3} \\ &= w_1 h_1 + w_2 h_2 + \cdots + w_9 h_9 \\ &= \sum_{i=1}^{9} w_i h_i. \end{aligned} \tag{2.11}$$

Adding a bias term to Equation (2.11), we see that we can express the equations of the convolutional layer in the same way as we did for the layer of a fully connected layer in Equation (2.7),

$$\begin{aligned} a_{x,y}^{(l)} &= \sum_{i=1}^{9} w_i^{(l)} h_i^{(l-1)} + b^{(l)} \\ &= w^{(l)} \star h_{x,y}^{(l-1)} + b^{(l)}; \quad h_{x,y}^{(l)} = g^{(l)}(a_{x,y}^{(l)}), \end{aligned}$$

where $l = 1, 2, ..., L_c$, where L_c is the number of convolutional layers. If $l = 1$, $h_{x,y}^{(0)}$ is not the values of a hidden layer, but the values of the input image(s).

Padding

If parts of the kernel lies outside of the input array, the summation is undefined. This problem is solved by *padding*; adding either zeros or a value around the borders of the array. Typically, an image is either not padded (referred to as *valid* convolution), or padded such that the dimensions of the image is kept intact (referred to as *same* convolution).

Sliding window

Recall that the first computation in a neuron from a fully connected network is a weighted sum of the inputs. In a convolutional layer, the first computation in a neuron is a convolution. Combining all the neurons of a convolutional layer, the

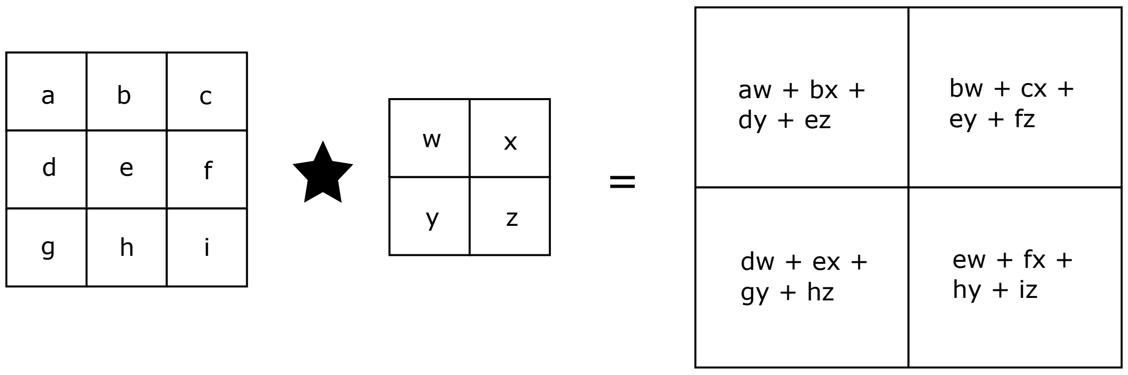


Figure 2.4: A 2-dimensional convolution of a 3×3 input by a 2×2 kernel. The output is a 2×2 feature map.

kernel looks like a window that slides over the image. As every kernel can be said to search for one specific feature of the image, the sliding window effect of the convolutional layer ensures that the same feature can be detected independently of location. Figure 2.4 is an illustration of a 2-dimensional convolution in a neural network, where the values of the output elements are described by Equation (2.10). Here, the kernel is restricted to be inside the image, a valid convolution.

Stride

The stride of a convolution is the number of increments by which the receptive field is moved during the convolution. In Equation (2.10), the stride is one. Using a stride larger than one is one way to reduce the amount of data. With a stride of two, the image resolution is reduced by one-half in each dimension, corresponding to the data amount in the image being reduced by three-fourths (Gonzalez and Woods, 2018). A stride larger than one can also be an alternative to subsampling, or pooling, which is discussed below.

ReLU

The rectifier activation function, defined as

$$g(x) = \max\{0, x\}, \tag{2.12}$$

is the recommended activation function for deep neural networks, i.e., neural networks where the number of hidden layers exceeds three (Glorot et al., 2011). A neuron with this activation function is referred to as a rectifier linear unit (ReLU). A motivation for using the rectifier activation function, is the behavior of the derivative. As long as the neuron is active (does not output zero), its derivative will stay constant, enforcing learning. Other activation functions have struggled with vanishing and exploding gradients, gradients that either become small and keep decreasing,

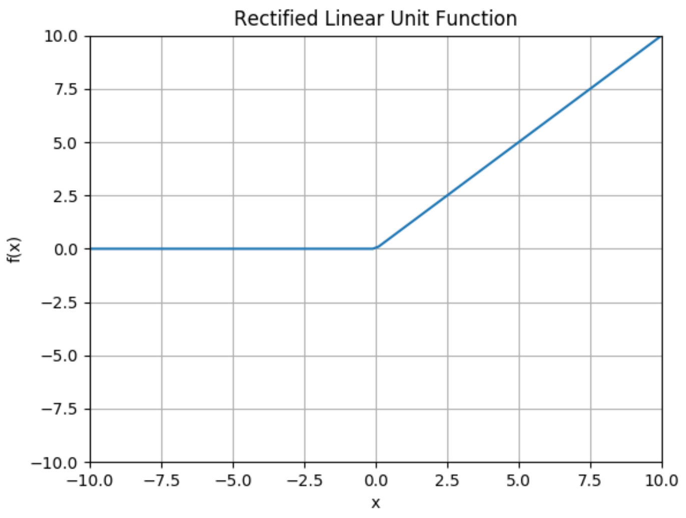


Figure 2.5: The rectifier function

or become too large and keep increasing. Using the rectifier function avoids this issue. In addition, the rectifier activation function allows for sparse representations in the network, as some of the neurons will output zero (Glorot et al., 2011).

Pooling

Pooling keeps the model less sensitive to small changes in the input. It replaces the output from the convolution with a statistical summary of the nearby outputs. For example, the max pooling operation gives the maximum of the rectangular neighborhood as output. This works as a noise suppressant, as small changes in the data will likely not impact the output of the maximum much. Typically, the neighborhood used in pooling is a 2×2 region, and these regions do not overlap. In the same way as for the convolutions, we can change the stride of the pooling to a constant $C > 1$, reducing the dimension of the output even more as well as the computational cost. See Figure 2.6 for an example of a 2×2 max pooling with a stride of $C = 2$.

2.2.3 Transposed convolutions

Transposed convolutions are a backwards pass of convolutions, and have been used for semantic segmentation, and for visualizing and understanding convolutional neural networks (Long et al., 2015; Zeiler and Fergus, 2014). A convolution without padding will produce an output with a smaller dimension than its input, while transposed convolutions increase the dimensions of the feature map. In a convolution, the output is a sum of the weighted inputs from the receptive field of the kernel.

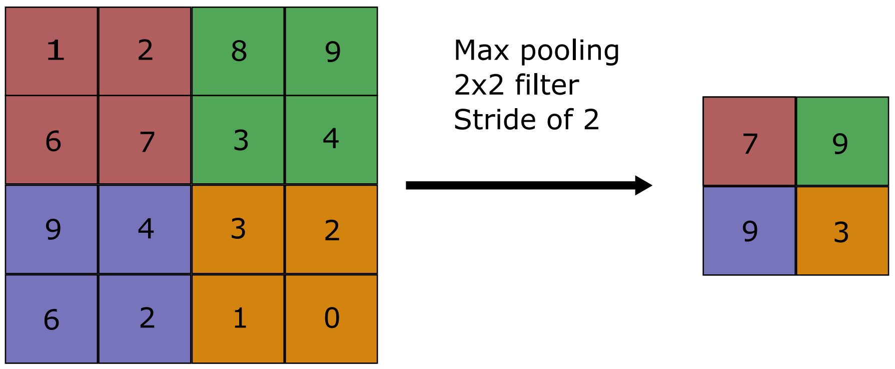


Figure 2.6: A 2 × 2 max pooling with a stride of 2. For each 2 × 2 region, the maximum value is stored in a new 2-dimensional array.

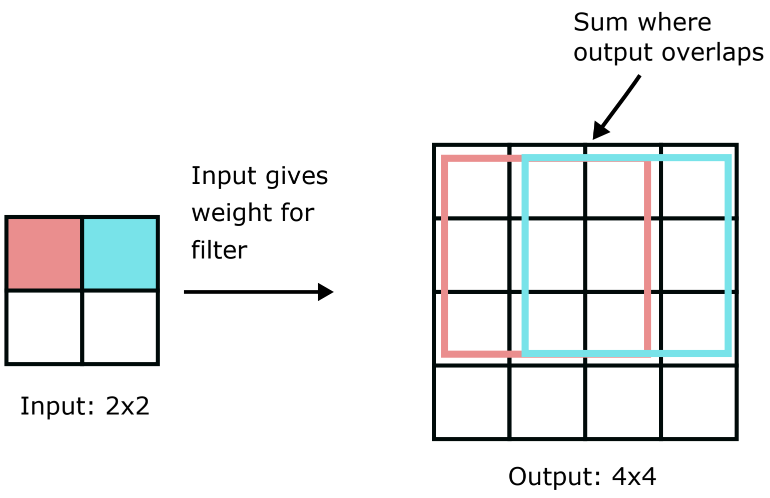


Figure 2.7: A transposed convolution. The input values gives a weight for the filter, also called the kernel, and it is then placed in its designated place in the output. Where values in the output overlap, they are summed.

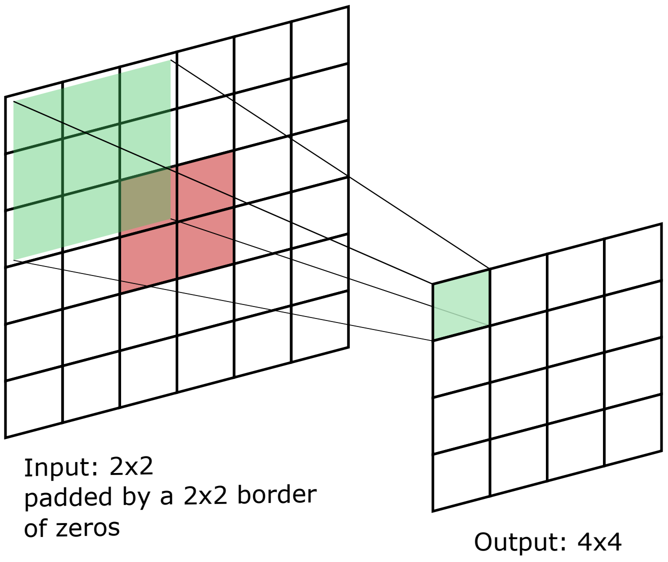


Figure 2.8: An alternative way to calculate the transposed convolution in Figure 2.7, by doing a convolution on a padded input. The input is padded such that the output is the desired dimension.

In the transposed convolution, the kernel is instead multiplied by each value of the input in turn, and then placed at the designated place in the output. If there is an overlap from values from before, these are summed, see Figure 2.7. The transposed convolution illustrated in Figure 2.7, can also be calculated by doing a convolution as described in Section 2.2.2, with a padding of 2, see Figure 2.8. Note that the size and stride of the kernel are the same.

Transposed convolutions with stride $C > 1$

The example above had a stride of 1. The transposed of a convolution with stride larger than 1 is done by adding *dilation* to the input. Dilation is done to widen the input even more, by inserting zeros between the input values. A transposed convolution with stride larger than 1 is therefore equal to doing a convolution on a dilated input with padding, see Figure 2.9.

2.2.4 Optimizing learning

Stochastic gradient descent

In the field of deep learning, the most common form of gradient descent is the stochastic gradient descent (SDG). SGD is an extension of the simple gradient descent that is mentioned in Section 2.2.2.

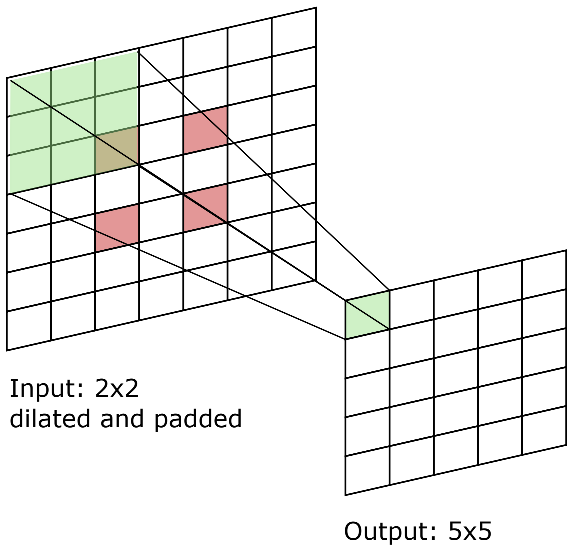


Figure 2.9: For transposed convolutions with step size > 1, zeros are inserted in between the input values. Here, the transposed of a convolution of a 5×5 input by a 3×3 kernel with step size 2, is calculated by a dilated 2×2 input with a 2×2 border of zeros convoluted by a 3×3 kernel with step size 1.

In machine learning, a larger training set is often better, but at the cost of computational expense (Goodfellow et al., 2016). The cost functions that are used are usually a sum over the training examples. If the training set is very large, calculating its derivative in order to take a gradient step can become a slow process.

Since the gradient is an expectation, we can estimate it using a small set of samples (Goodfellow et al., 2016). This small set of the training data, $\mathcal{B} = \{x_1, \ldots, x_{m'}\}$, is called a *minibatch*, and $x_1, \ldots, x_{m'}$ are drawn uniformly from the training set. The size of the minibatch, m' is typically small relative to the number of examples, m, in the training set. Keeping m' constant ensures that even though the training set grows, the computation time per gradient update is constant. Now, instead of calculating the cost function for the entire training set, we can calculate the cost function for each minibatch. Consequently, a gradient step can also be taken after each minibatch. The estimated gradient is

$$\mathbf{g} \leftarrow \frac{1}{m'} \nabla_{\mathbf{w}} \sum_{i=1}^{m'} E(\hat{y}_i, y_i), \tag{2.13}$$

where $\nabla_{\mathbf{w}}$ is the gradient with respect to $\mathbf{w}$. The SGD update rule can be expressed as

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon \mathbf{g}$$

Momentum

Learning with SDG can be slow, and the method of momentum (Polyak, 1964) can be used to speed up the learning. To compute a gradient step, the momentum algorithm makes use of not just the value of the gradient for this minibatch, but also the gradients from previous minibatches. Let $\mathbf{v}$ be the exponentially decaying average of the previous gradients, and let γ be the hyperparameter that decides how quickly the past gradients contribution should exponentially decay. The update rule can now be written as

$$\mathbf{v} \leftarrow \gamma \mathbf{v} - \epsilon \mathbf{g},$$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{v},$$

where $\mathbf{g}$ is defined in Equation (2.13) (Goodfellow et al., 2016). Now, the size of the gradient step depends on how aligned and how large the previous gradients are. The greatest step size is achieved if they all point in the exact same direction (Goodfellow et al., 2016).

2.2.5 Batch normalization

Another method motivated by the difficulty of training deep neural networks, is the method called batch normalization (Ioffe and Szegedy, 2015). As discussed in Sections 2.2.1 and 2.2.4, an update of a parameter in a layer is found by calculating the gradient, assuming that all the other layers do not change. In practice, the layers are all updated simultaneously. This can lead to unexpected results, as the layers are composed together (Goodfellow et al., 2016). A change of the input distribution of a layer, due to the change in the network parameters, is termed a *Internal Covariate Shift* (Ioffe and Szegedy, 2015). Batch normalization is a method constructed in order to avoid internal covariate shifts, by normalizing the inputs of the hidden layers in a neural network.

Let $\mathcal{B}$ be the minibatch of activations x, $\mathcal{B} = \{x_1, ..., x_{m'}\}$. These will be the input of the next layer in the network. Batch normalization is done by calculating the mean, $\mu_{\mathcal{B}}$, and variance, $\sigma^2_{\mathcal{B}}$, of the minibatch,

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m'} \sum_{i=1}^{m'} x_i,$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m'} \sum_{i=1}^{m'} (x_i - \mu_{\mathcal{B}})^2.$$

Each x_i is then standardized,

$$\hat{x_i} \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \zeta}},$$

where the ζ is a small positive value, added to avoid the undefined gradient when the standard deviation is zero (Goodfellow et al., 2016). This normalization has been shown to speed up convergence, even if the features are not decorrelated (LeCun et al., 1998). To make sure that the transformation does not change what the layer can represent, two parameters are introduced for each activation; γ and β. These scale and shift the normalized activation,

$$y_i \leftarrow \gamma \hat{x_i} + \beta,$$

and ensure that the network still has its representation power. γ and β are learned along with the rest of the weights of the network during training.

2.2.6 Cost function

A neural network for classification defines a distribution $p(y|x;\theta)$, and to optimize the model, the principle of maximum likelihood is used. The cost function will then be the negative log-likelihood, which can be described as the cross-entropy between the training data and the predicted data (Goodfellow et al., 2016). To understand cross-entropy, we first introduce the concepts entropy and Kullback-Leibner divergence.

Entropy

Let x be a random variable with distribution p. The entropy of x, which will be denoted by $\mathrm{H}(p)$ here, is a measure of the uncertainty of x. This is defined as

$$\mathrm{H}(p) = -\sum_{k=1}^{K} p(x=k) \log_2 p(x=k),$$

when x is a discrete variable with K states. When the log base 2 is used the units of entropy are bits. Given a K-ary random variable, the maximum entropy is obtained when $p(x=k) = 1/K$. The minimum entropy (an entropy of zero) is obtained if the function puts all its mass in one state (Murphy, 1993). Zero entropy means that the distribution has no uncertainty.

Kullback-Leibner divergence and cross-entropy

When training a neural network model, the aim is to make it predict the best approximation to the actual probability distribution of y. To use entropy to achieve this, we need some way of measuring how well the model predicts the distribution. The Kullback-Leibner (KL) divergence is a way to measure dissimilarity of two probability distributions. Let the two distributions be denoted by p and q, then the KL divergence is expressed as

$$\mathrm{KL}(p||q) = \sum_{k=1}^{K} p_k \log(p_k/q_k).$$

This can be rewritten as

$$\mathrm{KL}(p||q) = \sum_{k=1}^{K} p_k \log q_k - \sum_{k=1}^{K} p_k \log p_k = -\mathrm{H}(p) + \mathrm{H}(p,q),$$

where the last term, $\mathrm{H}(p,q)$ is referred to as the cross-entropy. As $\mathrm{H}(p)$ is the entropy of x with distribution p, the KL divergence is number of bits you need in addition when encoding the data using the distribution q instead of p. The cross-entropy is

$$\mathrm{H}(p,q) = -\sum_{k=1}^{K} p_k \log q_k, \tag{2.14}$$

and is the average number of bits needed to encode data coming from the distribution p, when q is used to encode the data (Murphy, 1993).

Minimizing cross-entropy with respect to q will be the same as minimizing the KL divergence with respect to q, as q is not present in the first term of the KL divergence. In other words, minimizing the cross-entropy will minimize the dissimilarity of the two distributions. When cross-entropy is used as a cost function, the distribution p will be the output vector $\mathbf{y}$, while the distribution q is the models predicted output, $\hat{\mathbf{y}}$, for a single input $\mathbf{x}$. As $\mathbf{y}$ is an one-hot vector, looking at Equation (2.14), we see that only the positive class contributes to the loss.

2.2.7 Output unit and the softmax function

The choice of cost function is connected to the choice of output unit. How the output is represented determines the form of the cost function.

We will here describe the softmax unit, as the model used in this thesis will be a multi-class classifier. This is the common choice whenever we want to predict a probability distribution over a discrete variable with n possible values. The output of the unit is the vector $\hat{\mathbf{y}}$, where $\hat{y}_i = P(y = i|x)$. Every $\hat{y}_i$ is between 0 and 1, and the vector sums to 1. As usual for a unit in the network, the weighted input of the unit is summed,

$$\mathbf{a} = \mathbf{w}^T\mathbf{x} + \mathbf{b},$$

where $\mathbf{b}$ is the bias of the unit. To obtain the desired $\hat{\mathbf{y}}$, the softmax function exponentiates and normalizes the components of the output layer,

$$\hat{y}_i = \text{softmax}(\mathbf{a})_i = \exp(a_i) / \sum_{j=1}^{n} \exp(a_j).$$

The exponential of the softmax can be undone by the log-likelihood,

$$\log \hat{y}_i = \log \text{softmax}(\mathbf{a})_i = a_i - \log \sum_{j=1}^{n} \exp(a_j). \tag{2.15}$$

From Equation (2.15), it is clear that the input a_i will directly contribute to the cost function. When the log-likelihood is maximized, the first term of the equation above will be increased, while the second term will be decreased. Further, from the second term we see that the most incorrect prediction will be the one that is penalized the most. If the correct prediction has the highest value, then $\log \text{softmax}(\mathbf{a})_i$ will roughly cancel. The training cost will then be small for that classification.

Usually, the negative log likelihood is minimized, instead of maximizing the log likelihood. Changing the sign of Equation (2.15) results in the loss function $\mathcal{L}$,

$$\mathcal{L} = -\log \hat{y}_i = -a_i + \log \sum_{j=1}^{n} \exp(a_j), \tag{2.16}$$

which is equivalent to taking the cross-entropy of $H(\mathbf{y}, \hat{\mathbf{y}})$, where $\mathbf{y}$ is the one-hot vector with $y_i = 1$.

2.2.8 Performance Metrics

The confusion matrix and singular assessment metrics

The confusion matrix is used as a performance measure for classification models. Given a data set $\mathbf{X}$, a classification model maps every element x_i of $\mathbf{X}$ to a class. To keep it simple, we will describe a model whose output is limited to two classes, a positive class and a negative class. Let p and n denote the true positive and negative class x_i, and let p' and n' denote the predicted positive and negative class of x_i. Then if the true class of x_i is positive and the model correctly classifies it as such, it is called a true positive (TP). If the model instead classifies it as negative, it is called a false negative (FN). In the same way, if the true class of x_i is negative, and the model correctly classifies it as negative, it is called a true negative (TN). While if it is classified as positive, it is called a false positive (FN). This can be described by a two-by-two matrix, called a confusion matrix, or a contingency table. See Table 2.1 for an example.

Table 2.1: The 2-by-2 confusion matrix

		True class	
		p	n
Predicted class	p’	TP	FP
	n’	FN	TN
	Total	P	N

There are several metrics we can compute from the confusion matrix. Firstly, the diagonal of the confusion matrix gives the count of correctly classified elements, while the off-diagonal counts the misclassified elements. A frequently used metric calculated from the confusion matrix is the *accuracy*. Defined as

$$\text{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}}, \tag{2.17}$$

where $P = |p|$ and $N = |n|$, it tells us how close our model is to a perfect classification. However, accuracy can be misleading. If 95% of the elements of $\mathbf{X}$ are positive, leaving the remaining 5% negative, the model would achieve a 95% accuracy simply by classifying all elements of $\mathbf{X}$ as positive. An accuracy of 95% looks good, but if we are interested in finding the negative elements, the model is useless.

Due to this weakness of the accuracy metric, other performance measures are used by researches to assess classification models. Precision, recall and F_1-score are defined as

$$\text{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{2.18}$$

$$\text{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{2.19}$$

$$\mathrm{F}_1\text{-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

We can interpret precision as a measure of exactness, how many of the elements predicted as positive are correct (He and Garcia, 2009). While recall can be seen as a measure of completeness, how many of the positive elements did the model classify correctly. By inspecting the equations for precision and recall, we see that precision is distribution dependent, but recall is not. However, measuring performance based on recall alone is unsatisfactory, as it tells us nothing about how many examples are incorrectly labeled as positive (He and Garcia, 2009). Similarly, precision does not provide information about how many of the positive labels are incorrectly labeled as negative, but it does capture the change in false positives. Together, they can provide a good measurement of performance, especially for imbalanced data sets. One way of combining the two is the F_1-score, the harmonic mean of precision and recall. The F1-score ranges from 0 to 1, with 1 representing the best score, and 0 the worst. This measure is still sensitive to data distributions, but provides more insight to the model functionality than the accuracy metric (He and Garcia, 2009).

Precision-recall (PR) curves

Using the definition of precision (2.18) and recall (2.19), the PR curve is defined by plotting the precision rate over the recall rate. Models such as neural networks usually output scores, by setting a threshold that determines what qualifies as a high enough score, this output can be transformed to a discrete value representing a class. This threshold can also be thought of as how confident the model is in its prediction. Applying a range of thresholds will create a line of points, a curve, in the PR space. A good model of the data will have a curve close to the upper right corner in PR space. This indicates a high value for both precision and recall, which will give a high F_1-score.

Chapter 3

Materials and Methods

Two types of fish schools are simulated, and then the neural network's ability to distinguish the two is evaluated. The two types of schools are distinguishable by two features; relative frequency response and their shape. By interchanging these two features, and observing the model's performance, we will determine if the features impact performance differently. In order to achieve this, we perform three simulations (see Table 3.1).

Table 3.1: The different combinations of relative frequency response and shape that make up the simulated training set and tests.

	Class	**Relative frequency response**	**Shape of schools**
Training and baseline test	1	$r_1(f)$	Spheroid
	2	$r_2(f)$	Ellipsoid
Shape test	1	$r_1(f)$	Ellipsoid
	2	$r_2(f)$	Spheroid
$\boldsymbol{r(f)}$ **test**	1	$r_2(f)$	Spheroid
	2	$r_1(f)$	Ellipsoid

The first simulation is a data set for training. This data set consists of echograms, and segmentation masks, where there are two types of fish schools present. A single fish school has either relative frequency response $r_1(f)$ or $r_2(f)$ (see Section 3.1.3). If it has relative frequency response $r_1(f)$, then it is shaped as a spheroid in the simulation model, and its class label in the segmentation mask is 1. If it instead has relative frequency response $r_2(f)$, it has the shape of an ellipsoid in the simulation model, and its class label is 2.

The features are then interchanged in the next two data sets, in order to test the model. The combination of relative frequency response and shape are now interchanged, see Table 3.1. First, the shape is switched in what we will call the shape

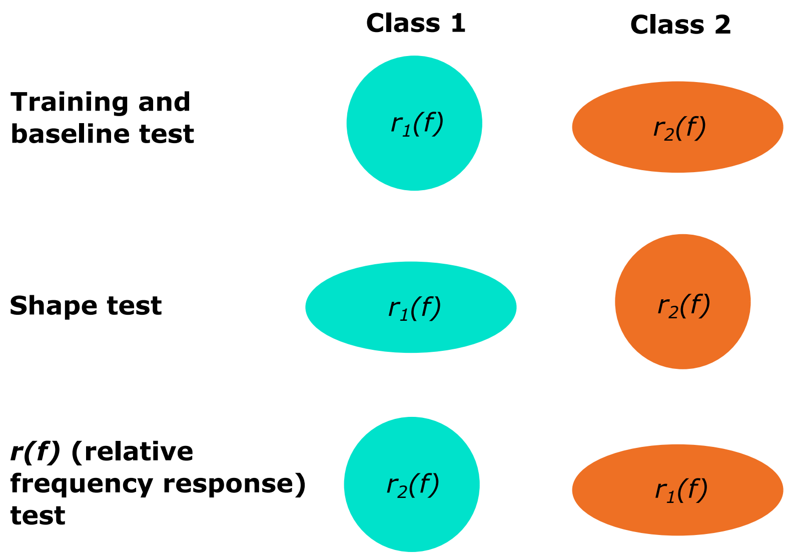

Class 1
Class 2
Training and baseline test
r1(f)
r2(f)
Shape test
r1(f)
r2(f)
r(f) (relative frequency response) test
r2(f)
r1(f)

Figure 3.1: The simulation setup. The training set and the first test have the same combination of features, where class 1 are schools shaped as spheroids with relative frequency response $r_1(f)$, and class 2 are schools shaped as ellipsoids with relative frequency response $r_2(f)$. For the echograms of the shape test the shape of the classes are interchanged from the baseline test. In the $r(f)$ test, the relative frequency response is interchanged. Note that in the figure the schools are drawn in the xy-plane of the simulation.

test, then the relative frequency response is switched in the second test set, which we will call the $r(f)$ test.

There are four different scenarios that will tell us the most about the model. If the model performance does not significantly drop on the shape test compared to the performance on the training data set, then our model ignores the shape of the schools when it classifies the schools. This would indicate that the classification is based on the relative frequency response of the fish species, as the rest of the characteristics are the same for both species. This should then be confirmed by checking the models performance on the echograms in the $r(f)$ test. If the model's performance on these echograms is significantly worse than for the training set, it confirms the importance of the relative frequency response.

A second scenario would be that the models performance drops significantly on the shape test, indicating that the shape of the schools are highly weighted for classification. If the predictions of the model is based solely on the shape of the schools, then this will be coupled with the model performing well on the $r(f)$ test, where relative frequency response is changed.

There is also the possibility that the model performance does not drop on either of the tests, or drops on both. The first case would indicate that the model weighs something other than relative frequency response and shape as most important, when classifying the echograms. If the model performance drops on both tests, both relative frequency response and the shape of the schools play a role in the classification.

The following chapter explains first how the simulation of the data sets is done, then the architecture of the neural network is described, along with its training scheme.

3.1 Simulating data

A data set is created using a model that simulates synthetic echograms from a multi-frequency echosounder. The echosounder is assumed to have collocated transmitter and receiver, also known as a monostatic echosounder, such as the Simrad EK60. The following descriptions and notations are based on Holmin et al. (2012).

3.1.1 Coordinate systems

The global coordinate system has its origin at the reference position of the research vessel, positive x is to the west, positive y is in the direction north, and positive z is vertically upwards.

Targets and transducer beams have their separate right hand Cartesian coordinate systems. The coordinate system of the single target has its origin at the center

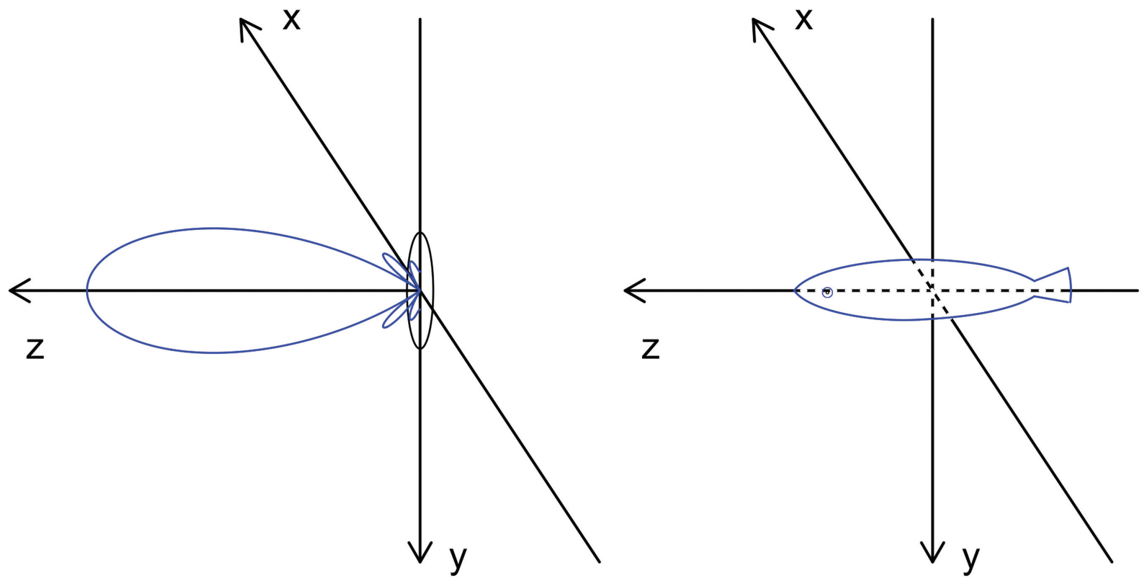


Figure 3.2: The coordinate systems of a single target, here represented by a fish, and the transducer beam. Reprinted with permission from A. J. Holmin et al., "Simulations of multi-beam sonar echos from schooling individual fish in a quiet environment", *The Journal of the Acoustic Society of America* 132.6. Copyright 2012, Acoustic Society of America.

of mass of the target, with z along the heading of the target, x axis is parallel to the sea surface, and the positive y axis is vertically downwards. For the transducer beams, the origin is at the transducer face, positive z is along the direction of the beam, the x axis is parallel to the sea surface, and positive y is in the negative vessel direction. A school of fish follows the global coordinate system. When discussing the simulation, we will use spherical coordinate systems.

The position of a target, (r, θ, ϕ), is defined by the range r, the azimuth angle θ, and the elevation angle ϕ, in the spherical coordinate system of a transducer beam. In the same way, the position of the transducer in the spherical coordinate system of the target is (r', θ', ϕ'). In our case, we will work with circularly symmetrical beams, making the azimuth angle redundant, leaving us with the target position (r, ϕ) and transducer position (r', ϕ'). Additionally, the movement of the transducer between transmission and reception of the sound waves is assumed to be negligible, meaning that $r' \approx r$. The transformation between the coordinate systems is explained in detail in Holmin et al. (2012).

3.1.2 Model of fish target strength

As mentioned in Section 2.1.5, the backscattering cross section at a distance r from the sound source is the measure of backscattered intensity at 1 m relative to incident intensity. Let σ_0 be the maximum backscattering cross-sectional area, obtained if the fish is perpendicular to the direction of the sound wave. The parameter σ_0 is dependent on the frequency of the sound wave f measured in kHz, and target size L measured in centimeters. To calculate σ_0 we will use the estimated target strength of herring at $f_0 = 38$ kHz,

$$\mathrm{TS} = 10\log_{10}\sigma_{0,f_0}(L) = 20\log_{10}L - 71.2,$$

recommended by ICES (ICES, 2008). Solving for $\sigma_{0,f_0}(L)$ we get the expression

$$\sigma_{0,f_0}(L) = L^2 10^{-7.12}.$$

To find the expression for the maximum backscattering coefficient of any frequency f, Holmin used results from five herring surveys done near Norway to fit a model to the ratio $\sigma_{0,f}(L)/\sigma_{0,f_0}(L)$. We recognize this ratio as the expression for relative frequency response, discussed in Section 2.1.5. The model $(f/f_0)^\gamma = \sigma_{0,f}(L)/\sigma_{0,f_0}(L)$ was fitted using the least-squares method, and estimated $\gamma = -0.4$. Using this model, the expression for $\sigma_{0,f}(L)$ becomes

$$\sigma_{0,f}(L) = (f/f_0)^{-0.4}L^2 10^{-7.12}. \tag{3.1}$$

If the target is not perpendicular to the direction of the sound wave, the backscattering cross section at an angle ϕ' is expressed in the simulation model as

$$\sigma_{bs,f}(\phi', L) = \sigma_{0,f}(L)\eta_{\phi'}B_l(\phi'),$$

where $\eta_{\phi'} \in [0,1]$ is the orientation factor of the target at the angle ϕ', and B_l is the beam pattern of a target simulated by the sinc function of the product $kl\phi/2$,

$$B_l = \mathrm{sinc}(kl\phi/2) = \frac{\sin(kl\phi/2)}{kl\phi/2},$$

where l is the measure corresponding to swimbladder length, and $k = 2\pi f/c$, where f is the frequency of the wave and c is the speed. The orientation factor $\eta_{\phi'}$ is frequency independent, and calculated assuming a cylinder rounded at both ends by hemispheres as a model of the target.

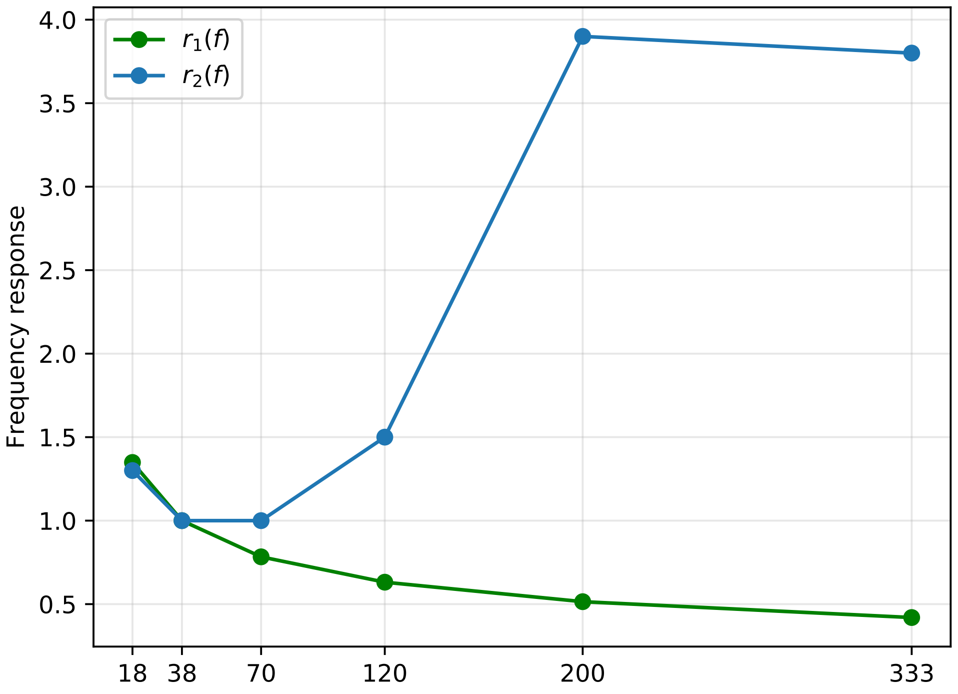


Figure 3.3: The approximated relative frequency response of herring $r_1(f)$, as calculated by Holmin et al. (2012), and the relative frequency response of mackerel $r_2(f)$, as reported by Fernandes et al. (2006)

3.1.3 The modeled relative frequency response

In the simulation model, two relative frequency responses are used. These are based upon the relative frequency response of herring, as calculated by Holmin et al. (2012), and the relative frequency response of mackerel, as reported by Fernandes et al. (2006), see Figure 3.3. It is important to note that the goal is not to simulate herring schools, or mackerel schools, but rather to use realistic values for the relative frequency response.

In order for the two types of fish schools to have different relative frequency responses, we let the maximum backscattering cross-sectional area $\sigma_{0,f}(L)$ differ for the two types. Recall the expression for $\sigma_{0,f}(L)$ from Section 3.1.2,

$$\sigma_{0,f}(L) = (f/f_0)^{-0.4} L^2 10^{-7.12}.$$

Recall that $(f/f_0)^{-0.4}$ was fitted from the ratio $\sigma_{0,f}/\sigma_{0,f_0}$, which is the relative frequency response for a single target, discussed in Section 2.1.5. Therefore, we

Table 3.2: Relative frequency response for Atlantic mackerel

Frequency (kHz)	$\boldsymbol{r_2(f)}$
18	1.3
38	1.0
70	1.0
120	1.5
200	3.9
333	3.8

let $r_1(f)$ be defined as

$$r_1(f) = (f/f_0)^{-0.4}.$$

Rewriting the expression for $\sigma_{0,f}(L)$ we have

$$\sigma_{0,f}(L) = r_1(f)L^2 10^{-7.12}.$$

The second relative frequency response we will use is based on the values for the relative frequency response of Atlantic mackerel, found in the SIMFAMI report (Fernandes et al., 2006, p. 50). Note that the report lists the relative frequency response for 364 kHz, and not 333 kHz, however we use the reported relative frequency response of 364 kHz as an approximation for 333 kHz. The reported relative frequency responses are listed in Table 3.2. Using this table, we can express the maximum backscattering coefficient for this case as

$$\sigma_{0,f}(L) = r_2(f)L^2 10^{-7.12}.$$

The two types of relative frequency response can be seen in Figure 3.3.

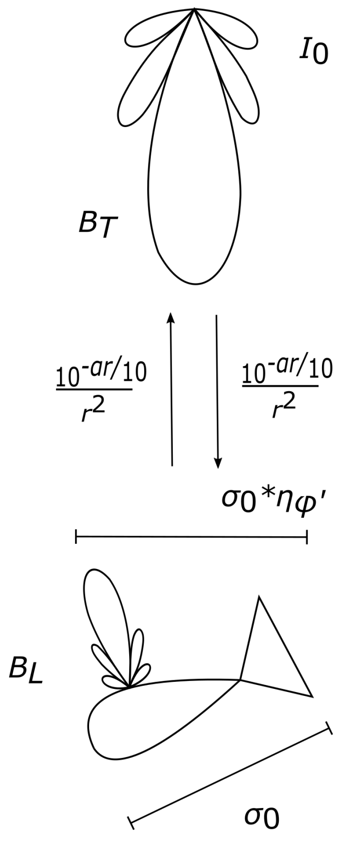
3.1.4 Intensity received from a single target

The received sound intensity, I_{rec}, from a single target at distance r is described as

$$I_{rec} = I_0 \frac{10^{\alpha r/5}}{r^4}[B_T(\phi)]^2 \sigma_{bs}(\phi'),$$

where I_0 is the initial intensity of the sound wave as it is transmitted by the transducer, $10^{\alpha r/5} r^{-4}$ is the two way loss of intensity as the sound wave propagates through the water, discussed in Section 2.1.3, $\sigma_{bs}(\phi')$ is the backscattering coefficient of a target at an angle ϕ' described in Section 3.1.2, and B_T is the beam pattern for emission and reception, modeled by the same circular piston

$$B_T(\phi; k, a) = \left(\frac{2J_1[ka\sin(\phi)]}{ka\sin\phi}\right)^2, \tag{3.2}$$



I_0
B_T
$\frac{10^{-ar/10}}{r^2}$
$\frac{10^{-ar/10}}{r^2}$
$\sigma_0 * \eta_{\varphi'}$
B_L
σ_0

Figure 3.4: An illustration of how the intensity received from a single target is modeled in the simulation model. The received intensity is calculated by multiplying the initial intensity by the loss of intensity due to beam spreading and absorption, the backscattering coefficient of the target at an angle ϕ', and the beam pattern for emission and reception.

where a is the radius of the circular piston and k is the wave number defined in Section 3.1.2, J_1 is the Bessel function of the first kind. A detailed discussion of Equation (3.2) can be found in (Kinsler et al., 2000).

3.1.5 Multiple targets and echograms

The domain of the simulation is a box that spans 200 meters out from the vessel in both negative and positive x-direction, and the depth of the box is 200 meters in the negative z-direction. The vessel moves in the positive y-direction determined by the number of pings set by the operator. Schools, aggregations of fish that swim in the same direction, are simulated by ellipsoidal clusters of uniformly distributed targets. These are randomly placed inside the domain of the transect. The schools are not tilted, however the average tilt of the individual targets inside is 20 degrees.

The heading of the school, rotation around the z axis, is sampled from an uniform distribution between (-π, π). Each target's individual heading is also perturbed, done in such a way that the average difference between the heading of the school and the heading of each individual target is 20 degrees.

The echograms produced by the simulation model are 2-dimensional arrays of pixels, where each column of pixels correspond to a transmission from the transducer. Each of these pixels contains the sum of the intensities received from each target, l, within the sampling volume, called voxel, at the depth of the pixel. If j denotes the sampling volume, then the sum is given as $I_{rec,j} = \sum_l I_{rec,j,l}$, where $I_{rec,j,l}$ is the intensity of target l in sampling volume j. As the simulation is done in a 3-dimensional domain, the voxel is a curved disc with constant thickness and increasing radius along the beam direction.

The randomness due to constructive or destructive interference is accounted for by considering $I_{rec,i}$ to be the mean of an exponentially distributed variable. Rayleigh showed that the amplitude of the sum of many sine waves that have the same frequency and random phases, are Rayleigh distributed. The probability density function (PDF) of the amplitude can therefore be expressed as

$$f_A(x) = \frac{x}{\sigma^2} \exp(-x^2/2\sigma^2),$$

where $\sigma^2 = \sum_l a_l^2/2$, where a_l is the amplitude of the lth sine wave. Holmin et al. (2012) states that this implies that the intensity I is exponentially distributed with mean equal to the sum of the individual intensities. See Holmin et al. (2012) for further details.

3.1.6 The shape of the schools

The schools in the simulation are either shaped as ellipsoids or shaped as spheroids. These shapes were chosen in order to create a clear distinction between the schools.

Schools in the simulation are clusters of targets, where the size of the targets, L, is drawn from a Normal distribution with mean 32 centimeters and standard deviation 2 centimeters. All schools have a volume $V = 10^5 = \frac{4}{3}\pi abc$, where a is the width, b is the length and c is the height of the school. As the schools are not tilted upwards or downwards, the heading of the school is parallel with the sea surface. Equivalently, a and b are parallel to the sea surface, while c is perpendicular to the sea surface. If a school has the shape of a sphere, then a, b, and c are approximately the same. Schools shaped as ellipsoids have instead the same width and length, but with a height that is 1/5 of these.

3.1.7 The segmentation mask

In order to train a neural network on the simulated data, a segmentation mask is created for each echogram containing the ground truth for each pixel. The following section describes the method for creating the segmentation masks.

For each voxel, a search for nearby schools is done within a given radius. A school is an ellipsoid with length a, width b and height z. At depth d of the voxel, a horizontal cross section of the ellipsoid has length a', width b', and center $c_s = (x_s, y_s)$. Let $\mathbf{D}$ be the vector from c_s to the center of the voxel, $c_v = (x_v, y_v)$,

$$\mathbf{D} = c_v - c_s.$$

$\mathbf{D}$ is then used to find the distance, d_e, from c_s to the voxel edge,

$$d_e = ||\mathbf{D}||_2 - r,$$

where r is the radius of the voxel. Let e be the closest point of the coxel edge to c_s, found by adding d_e to c_s,

$$e = (x_e, y_e) = c_s + d_e \cdot (\sin(\theta), \cos(\theta)),$$

where θ is the angle between the long axis of the cross section and $\mathbf{D}$. Then if e is inside the school ellipsoid the inequality

$$\frac{x_e^2}{a'^2} + \frac{y_e^2}{b'^2} \leq 1$$

will hold. The corresponding position in the segmentation mask is then marked as a constant $C > 0$ if there is a fish species present. If the inequality does not hold, meaning there are no fish species present, then the position in the segmentation mask is marked with a zero. Figure 3.5 shows parts of an echogram with a corresponding segmentation mask.

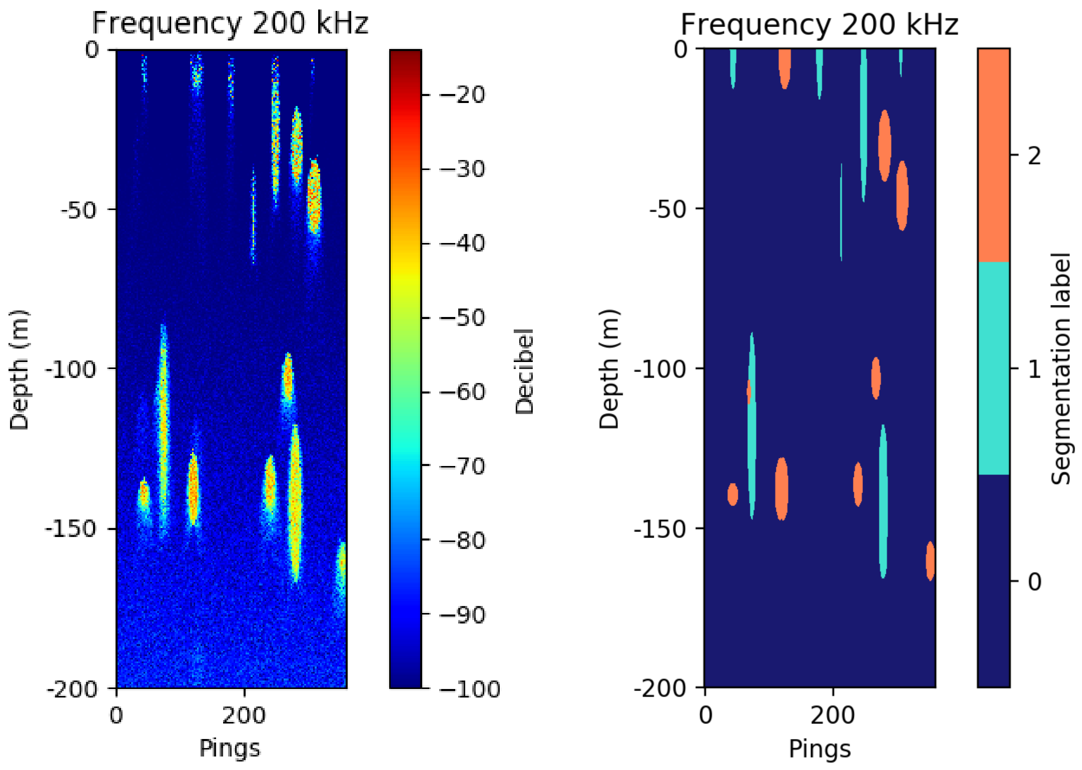


Figure 3.5: An example of an echogram of frequency 200 kHz and corresponding segmentation mask. The echogram belongs to the training set, therefore label 1 of the segmentation mask corresponds to fish schools shaped as spheroids and with relative frequency response $r_1(f)$. Note that these create more backscatter, as they have more targets perpendicular to the sea surface of the simulation. Label 2 corresponds to fish schools shaped as ellipsoids and with relative frequency response $r_2(f)$. Label 0 is background, or seawater. The segmentation labels are also referred to as class 0, 1 and 2 later.

Table 3.3: Sampling strategy for the training crops

Class	Probability	Description
0: Background	1/11	Crop from area without fish schools
1: Fish school	5/11	Crop with fish school of class 1 present
2: Fish school	5/11	Crop with fish school of class 2 present

3.2 Training a model

The model used for training and evaluation is based on the U-Net model, a neural network model first proposed for the purpose of segmenting blood cells (Ronneberger et al., 2015). Brautaset et al. (2020) showed that a slightly modified version of the U-Net can successfully classify echograms. The following sections describes how a similar model is trained to classify echograms.

3.2.1 The training set

In the simulation of the training data, we have simulated continuous echograms for the frequencies 18, 38, 70, 120, 200 and 333 kHz, see Figure 3.6. In training, these are cropped into patches of dimension $6 \times 256 \times 256$, where 6 is the number of frequencies, and 256 is the height and width of the patch, see Figure 3.7. A decibel transform is also applied to every value of the echogram, and a threshold is set at -75 dB and 0 dB, such that every value below or above the thresholds is set to -75 dB and 0 dB respectively.

The data set has a heavy class imbalance, as a natural consequence of the ratio between water and fish. Therefore, we need to ensure that the model is not overly exposed to "background" pixels. The crops are done in such a way that they include at least one fish school, or none at all. This way, we can sample the crops that includes a fish school more frequently than those without, seeing as there will also be "background" pixels in the ones with fish schools. The sampling strategy is listed in Table 3.3, the ratio between probabilities is similar to Brautaset et al. (2020).

The training data is divided into a training set, a validation set and a test set. The test set is unseen by the model until evaluation, which will be discussed in the next chapter.

3.2.2 Model architecture

The model used in this thesis will be the same as presented by Brautaset et al. (2020), with some minor adjustments. This is referred to as a fully convolutional network, meaning there are no fully connected layers, see Section 2.2.1. The model has two parts, a contracting part and an expansive part, as seen in Figure 3.8.

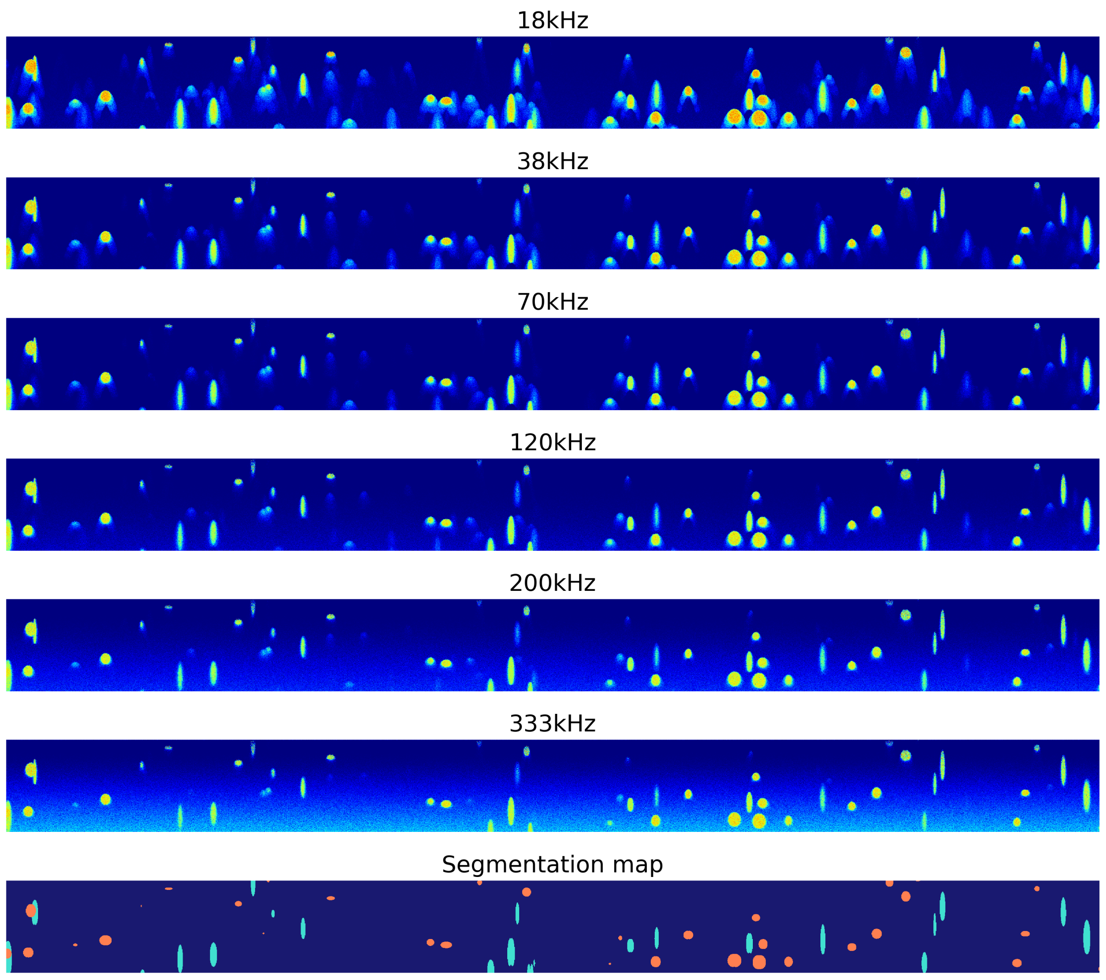
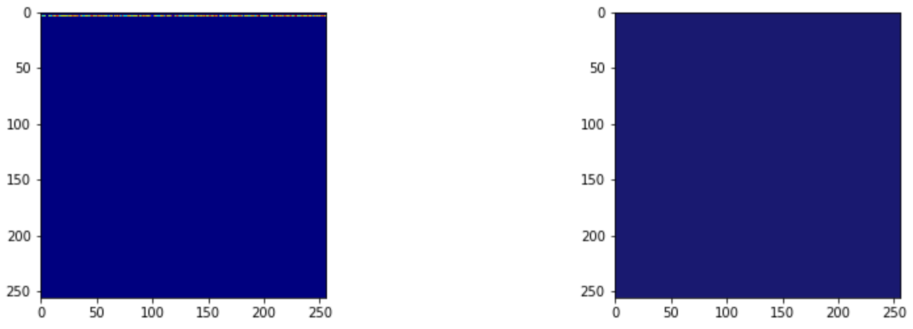
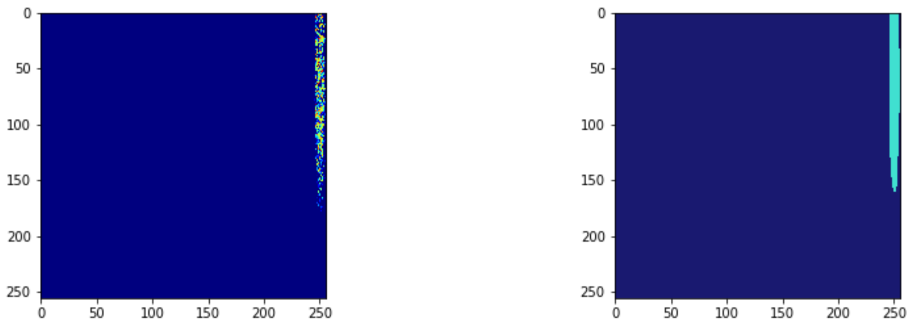
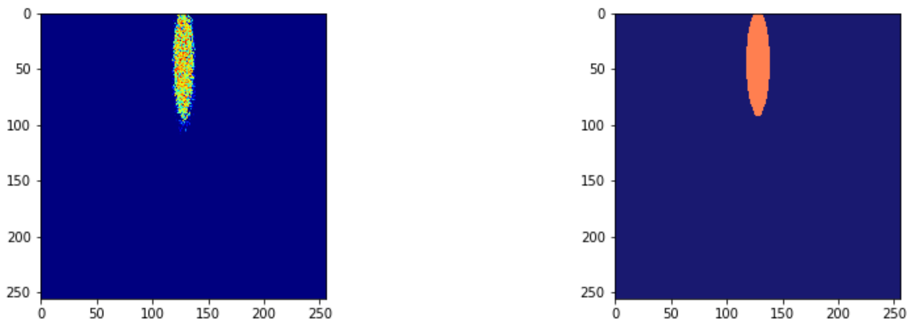


Figure 3.6: An echogram created by the simulation model. Each echogram is simulated for six frequencies: 18, 38, 70, 120, 200 and 333 kHz. The segmentation map is a 2-dimension array which contains the label of each pixel. Each pixel is either background, which has label 0, or a fish school, which is labeled 1 or 2 depending on its characteristics. Background pixels are colored in dark blue, while fish schools belonging to class 1 are colored cyan, and fish schools belonging to class 2 are colored orange.


(a) A crop with only class 0


(b) A crop with class 1 present


(c) A crop with class 2 present

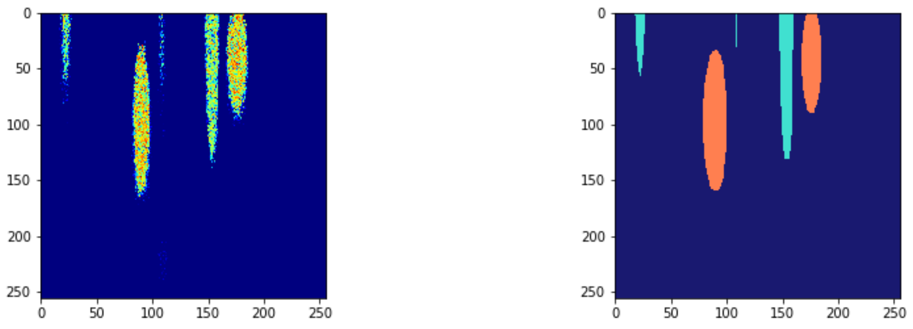

(d) A crop with class 1 and 2 present

Figure 3.7: Examples of training crops with corresponding true segmentation masks. Every crop has dimensions 6 × 256 × 256, where 6 is the number of frequencies simulated. Only one frequency is shown here.

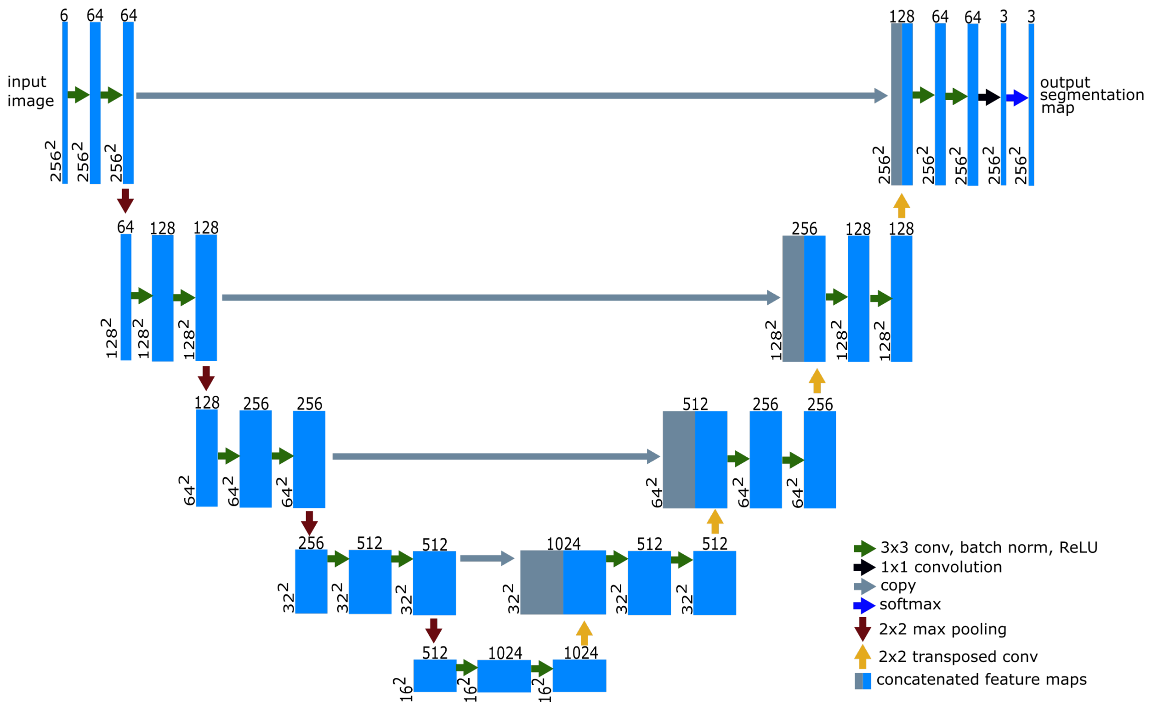


Figure 3.8: U-Net architecture

The input of the model has dimension $6 \times 256 \times 256$ as described in Section 3.2.1. Then the contracting part follows, with blocks of 3x3 same convolutions, each followed by a rectified linear unit (ReLU), see Section 2.2.2. The convolution layers are the ones that detect features of the data set, while ReLU is a neural network unit where the activation function is defined as $g(x) = \max\{0, x\}$. Brautaset et al. (2020) added a batch normalization layer between each convolutional layer and its subsequent activation function, and we follow this strategy. The batch normalization layer normalize the input of the hidden layers in the neural networks, and its benefit is discussed in Section 2.2.5. Each block is followed by a 2×2 max pooling operation with stride 2. As discussed under Section 2.2.2, this keeps the model less sensitive to small changes in the input, as well as reducing the data amount.

In the expansive part each block consists of layers with 3×3 convolutions as well, now followed by an transposed convolution to increase the patch resolution, described in Section 2.2.3. The input of each block is also concatenated with the feature map learned from the corresponding layer in the contracting path in order for the successive convolution layer to assemble a more precise output (Ronneberger et al., 2015). The last convolutional layer is a 1×1 convolution which reduces the number of channels, followed by a softmax layer that is used to create the predicted segmentation map, see Section 2.2.7. The softmax layer has three channels, one for each class, with the softmax value for each pixel of the input patch.

3.2.3 The training scheme

We follow the training scheme of Brautaset et al. (2020), training the model over 5,000 iterations with batches of size 16. The weights of the model are initialized from a random uniform distribution, and are optimized using stochastic gradient descent, with an initial learning rate of 0.01 and momentum set to 0.95, see Section 2.2.4. The high momentum was also recommended by Ronneberger et al. (2015), to ensure that a high number of the previously seen training samples contribute to the update of the current optimization step. Every 1,000 iteration, the learning rate is reduced by a factor of 0.5.

Brautaset et al. (2020) also use a weighted cross entropy loss, to further adjust for the class imbalance. The cross-entropy is done for each of the channels of the output layer. We will here describe how it is calculated for one channel. Rewriting Equation (2.16) from Section 2.2.7 to the 2-dimension case, we have

$$\mathcal{L}_{i,j} = -\log \hat{y}_{i,j} = -a_{i,j} + \log \sum_m \sum_n \exp(a_{m,n}),$$

where m and n span the dimensions of the layers of $\mathbf{a}$. This is the negative log likelihood of the element at position (i,j) of the softmax layer, equivalent to calculating the cross-entropy between $\mathbf{y}$ and $\hat{\mathbf{y}}$. To account for the class imbalance, each class of the echogram is given a weight, and the weighted cross-entropy loss is expressed as

$$\mathcal{L}_{w(y_{i,j})} = w(y_{i,j}) \left(-\log(\text{softmax}(a)_{i,j}) = -a_{i,j} + \log \sum_m \sum_n \exp(a_{m,n}) \right),$$

where

$$w(y_{i,j}) = \begin{cases} 1, & \text{if } y_{i,j} = 0 \\ 30, & \text{otherwise,} \end{cases}$$

where $y_{i,j}$ is the element at position (i,j) of the true segmentation mask. The choice of $w(y_{i,j})$ is based on Brautaset et al. (2020).

The patches are also randomly flipped about the vertical axis, as well as added random multiplicative noise to 5% of their pixels (chosen randomly) to increase the robustness of the model.

3.3 Testing the model

Three test sets are simulated, as described in Table 3.1. All three test sets have ten echograms each. Each of these echograms have dimensions $6 \times 1056 \times 2500$, where 6 is the number of frequencies, 1056 is the height of the echogram and 2500 is the length of the echogram. Figure 3.6 shows an example of a simulated echogram.

First, the model is tested on echograms similar to the ones it has been trained on. This gives a baseline performance of the model, which the performance on the echograms of the other tests will be compared to. The pixel accuracy is calculated, measuring how close the model is to perfect classification, as described in Section 2.2.8. The accuracy is reported for each of the fish school classes. The model's performance on classifying the background pixels is expected to be high due to the high number of pixels of this category, and will not be reported as it is not important for the objective of this thesis. The precision-recall curve is also calculated. Precision, recall and F_1-score give a better indication of performance for imbalanced data sets, as described in Section 2.2.8. Generally in echograms, there is a high amount of background compared to fish, and this is also true for the simulated data. From the precision-recall curve, the best F_1-score is found.

The pixel accuracy of the model on the echograms of the two other tests is calculated. To see if there is a difference in model performance between the tests when it comes to accuracy, the accuracy of the model on each class is compared between tests. This is done by testing if the mean accuracy is equal, i.e. if the mean accuracy of the model on class 1 from the shape test is equal to the mean accuracy of the model on class 1 from the baseline test. If there is no significant difference in mean, then the performance of the model on class 1 does not change when the shape of the school is changed. We will use the Wilcoxon rank-sum test, a non-parametric alternative to the two-sample t-test, to test for equal means.

The Wilcoxon rank-sum test is done by first creating a pooled sample, C, that consists of the two samples that will be tested. Let A be a sample of size n with mean μ_1, and B be a sample of size m with mean μ_2. Then, order C from the lowest value to the highest value, and rank from 1 to N, where $N = n + m$, such that the lowest value gets rank 1 and the highest value gets rank N. Now, the test statistic W is the sum of the ranks associated with observations from sample A (Devore and Berk, 2011). If these are associated to either the most of the smallest ranks, or the largest ranks, we would doubt that the means of the samples, μ_1 and μ_2, are the same.

Lastly, the precision-recall curves for the shape test and the $r(f)$ test are compared to the precision-recall curve obtained for the baseline test. Recall that an indication that the model is a good fit to the data, is a precision-recall curve close to the upper right corner of the PR space, see Section 2.2.8. The best F_1-score is calculated for the tests, and compared to the score the model obtained on the baseline test. A good F_1-score will be close to 1.

Chapter 4

Results

4.1 Baseline performance

The model is trained and validated on the training set, before it is tested on the echograms of the baseline test. In this test, the schools have the same combination of relative frequency response and shape as in the training set, i.e. class 1 are schools with relative frequency response $r_1(f)$ and are shaped as spheroids in the simulation, while schools belonging to class 2 are shaped as ellipsoids in the simulation and have relative frequency response $r_2(f)$. The performance of the model on this test is the baseline performance of the model, which the performance of the model on the other tests is compared to. Figure 4.1 shows an example of a classification based on the predictions from the model on a 6-channel echogram from the baseline test.

The pixel accuracy on this test is high, with a mean accuracy of 0.985 and 0.988 for class 1 and 2, respectively. The accuracy for each class is calculated using Equation (2.17), and the mean accuracy for each class of each test is listed in Table 4.1. We omit the accuracy of the background pixels, as this is not important for the objective.

Table 4.1: Accuracy of the model on each of the tests.

		Class 1	**Class 2**
Baseline test	**Mean accuracy**	**0.985**	**0.988**
	Standard deviation	0.009	0.007
Shape test	**Mean accuracy**	**0.830**	**0.947**
	Standard deviation	0.021	0.016
$r(f)$ test	**Mean accuracy**	**0.055**	**0.160**
	Standard deviation	0.026	0.052

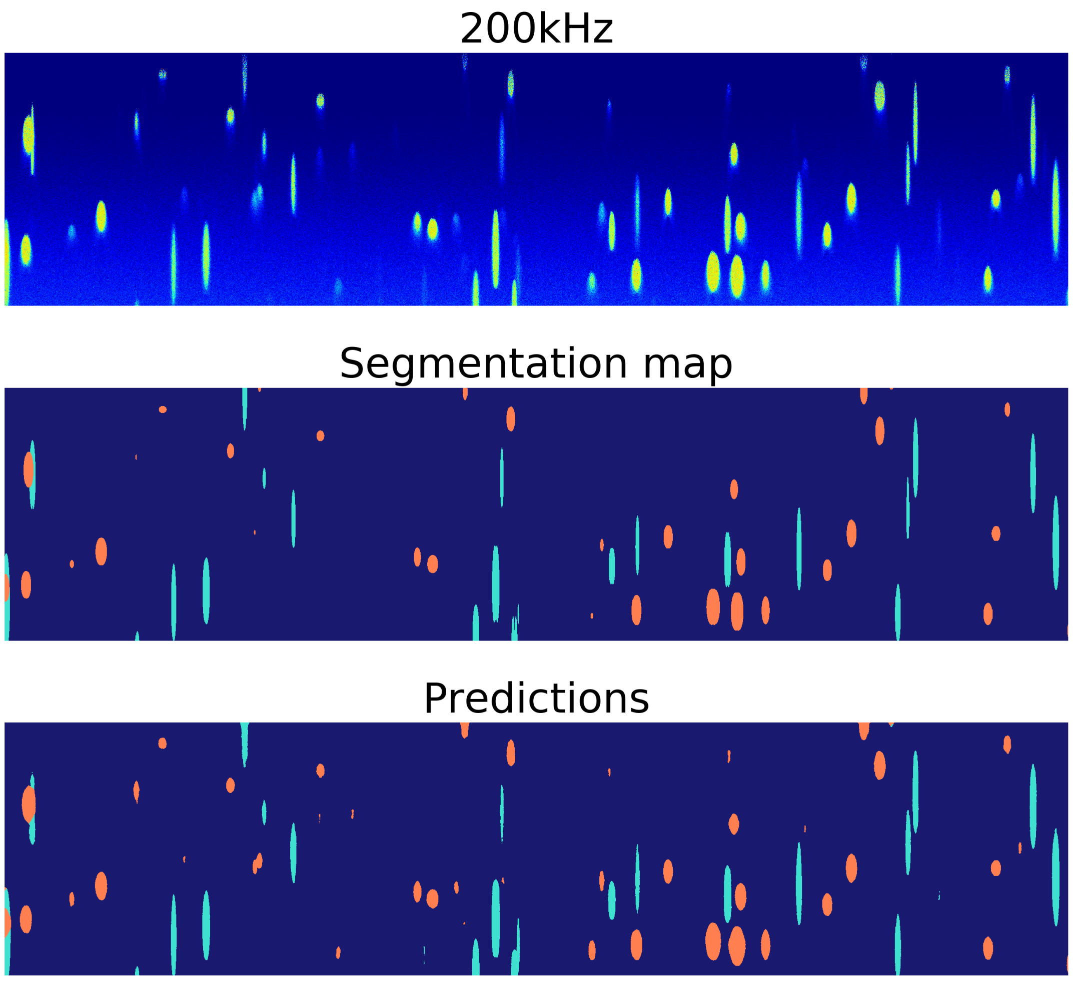
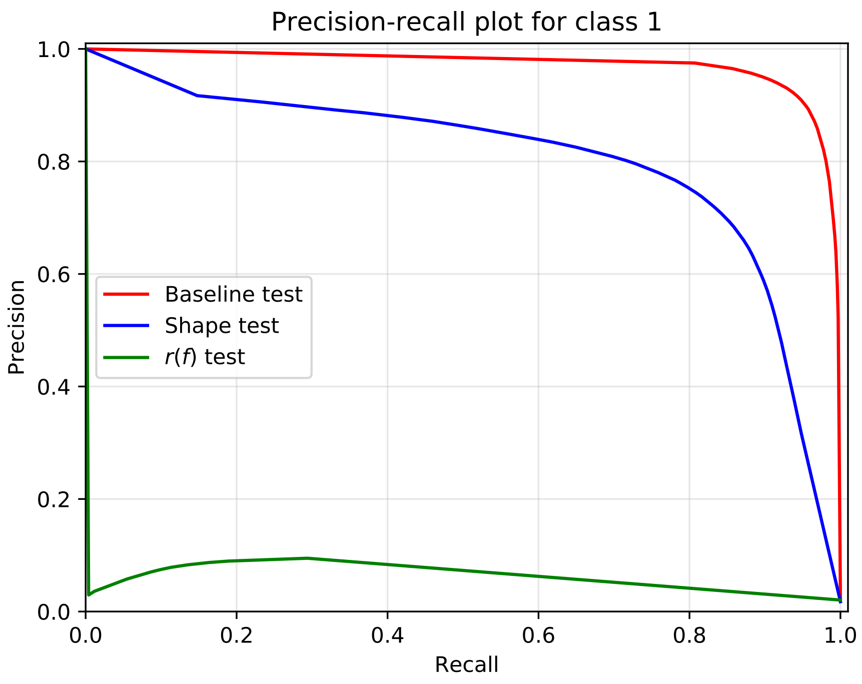


Figure 4.1: An example echogram from the baseline test, showing the 200 kHz channel, with corresponding true segmentation mask, and the segmentation mask from the model's prediction. Class 1, which includes schools with relative frequency response $r_1(f)$ and shaped as spheroids in the simulation, is colored cyan. Schools that belong to class 2, having relative frequency response $r_2(f)$ and shaped as ellipsoids in the simulation, are colored orange.

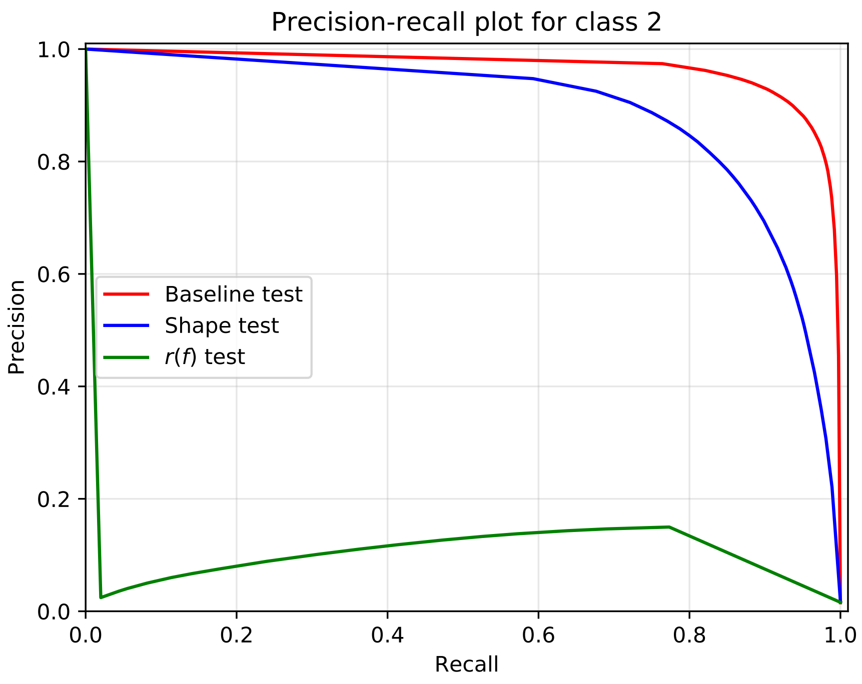
Table 4.2: Precision, recall and F_1-score for the three tests.

	Class 1			**Class 2**		
	Baseline test	**Shape test**	**$r(f)$ test**	**Baseline test**	**Shape test**	**$r(f)$ test**
Precision	0.922	0.749	0.095	0.907	0.835	0.150
Recall	0.937	0.804	0.294	0.930	0.811	0.773
F_1-score	0.930	0.775	0.143	0.918	0.823	0.251
Threshold	0.950	0.588	0.005	0.940	0.960	0.005

The precision-recall curve is calculated for the baseline test, and the best F_1-score is listed in Table 4.2, with its corresponding precision and recall. The best F_1-score obtained is 0.930 for class 1, at threshold 0.95, where the precision of the model is 0.922 and the recall of the model is 0.937. The high precision and recall indicates that the predictions from the model have both a low number of false positives and a low number of false negatives, see Section 2.2.8. For class 2, the best F_1-score is 0.918, found at threshold 0.940, obtained by a precision of 0.907, and a recall of 0.930.



(a) The PR-curves for class 1



(b) The PR-curves for class 2

Figure 4.2: PR-curves obtained from the three tests, where red, blue and green curves are the baseline, shape test and $r(f)$ test, respectively. The model obtains the best result for the baseline test, i.e., the data set with the same combinations of shape and relative frequency response as the training set. The performance of the model on the shape test, where the shape of the schools are interchanged, has dropped from the baseline test, with a slightly worse performance on class 1 than class 2. The worst performance is on the echograms from the $r(f)$ test, where the relative frequency response is interchanged between the two fish classes.

4.2 Testing the performance of the model

The first metric we will use to compare the performance of the model is the pixel accuracy. The mean pixel accuracy of the model on each of the tests is listed in Table 4.1. The performance of the model on the shape test and the $r(f)$ test is compared to the baseline test, by testing for equal distributions, as described in Section 3.3. Then, the precision-recall curve is investigated.

4.2.1 The shape test

The model is first tested on echograms where the classes have interchanged shape from what the model has been trained on. Figure 4.3 shows classification based on the model's predictions of an echogram from this test. The performance of the model on this test will be compared to the performance of the model on the baseline test. If the performance of the model drops significantly, then the shape of the schools affects the predictions of the model.

The results from the shape test indicate that changing the shape of the schools does affect the performance of the model. The pixel accuracy drops compared to the pixel accuracy on the baseline test, see Table 4.1. The drop in accuracy is more prominent for class 1, from 0.985 to 0.830, where the shape is changed from spheroids to ellipsoids, while for class 2 mean accuracy drops from 0.988 to 0.947.

The accuracy of the model on this test, and the accuracy of the baseline test, are tested for equal distributions, by the Wilcoxon rank-sum test, as described in Section 3.3. The accuracy of each test is divided into each class, such that there is done two tests. The results of the Wilcoxon rank-sum are significant for both classes ($W = 100$, $P < .001$), and we reject the hypothesis that the samples are from the same distribution.

The precision-recall curves for class 1, and class 2, of the shape test also indicates that changing the shape of the schools affect the performance of the model, see Figure 4.2. Both curves show that the model has a poorer performance than what it had on the baseline test. As with the pixel accuracy, the drop is most prominent for class 1. This is also shown in the F_1-scores, where it is 0.775, and 0.823, for class 1, and 2, respectively. The threshold for achieving the F_1-score is much lower for class 1, where it is 0.588, while for class 2 the threshold is 0.960, indicating that the model is still confident predicting class 2.

4.2.2 The $r(f)$ (relative frequency response) test

For the second test, which we will call the $r(f)$ test, the classes have interchanged relative frequency response compared to the baseline test. A classification based on

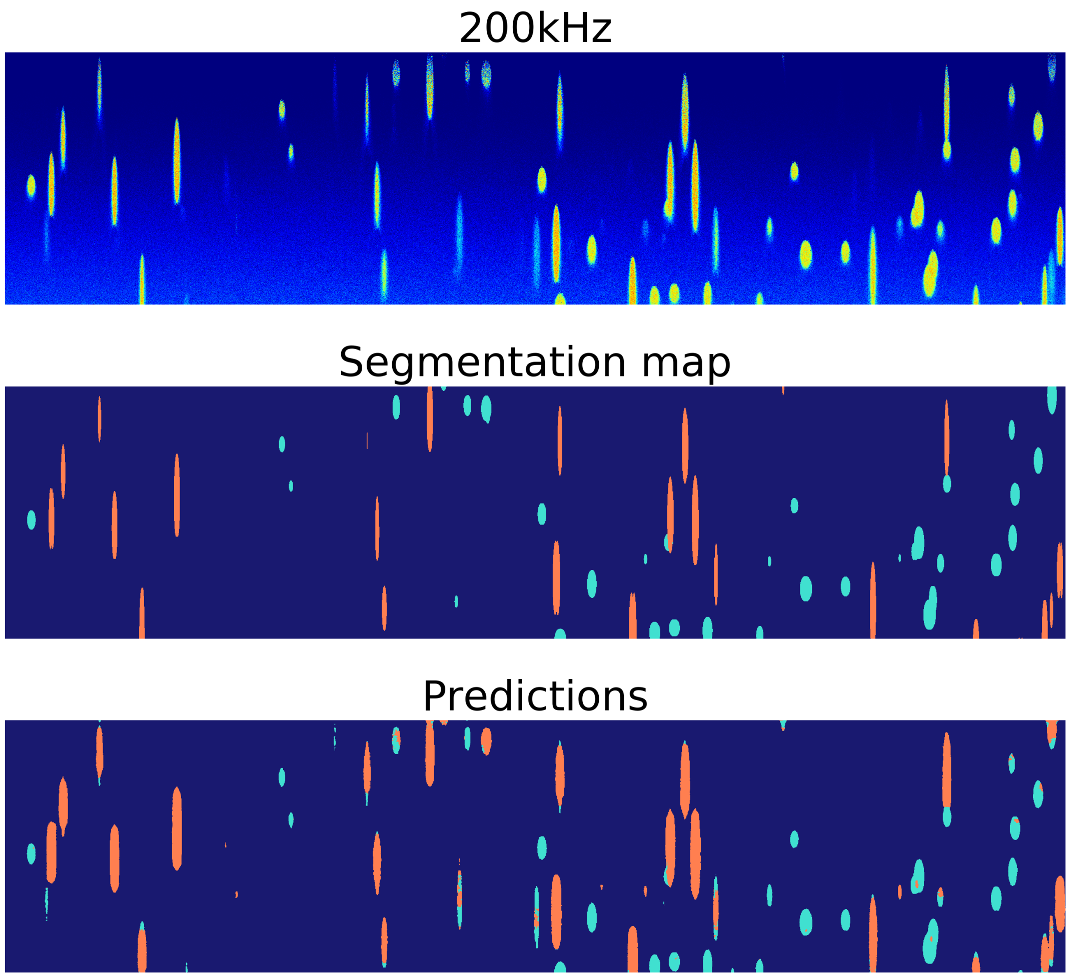


Figure 4.3: An echogram from the shape test, represented here by its 200 kHz channel, with its true segmentation mask and the segmentation map based on the model prediction. The shape of the classes are interchanged from the baseline test, i.e., class 1 is now shaped as an ellipsoid, and class 2 is shaped as a spheroid. In the segmentation map, pixels belonging to class 1 are colored cyan, while pixels belonging to class 2 are orange.

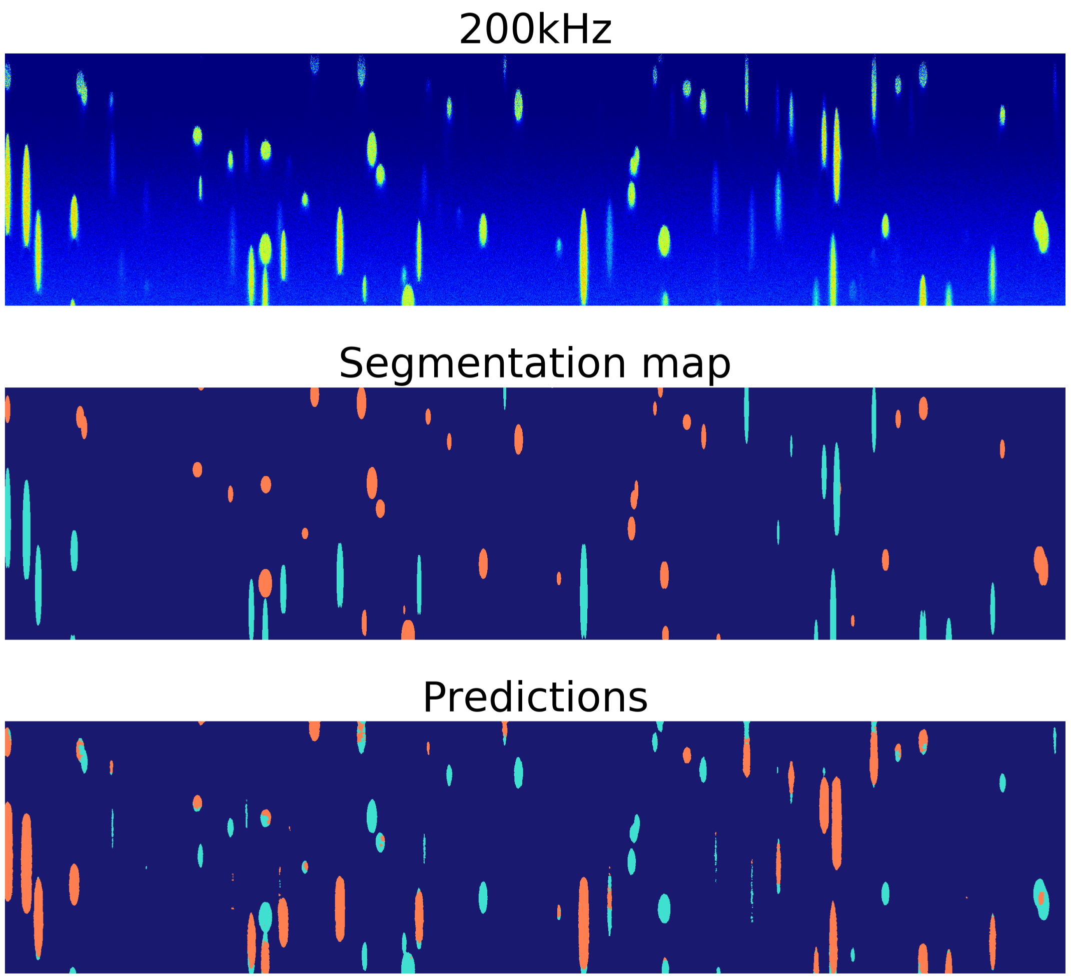


Figure 4.4: An echogram at frequency 200 kHz from the $r(f)$ test, with its true segmentation map and the segmentation map originating from the predictions of the model. The $r(f)$ test differs from the training set by having interchanged relative frequency response between the two fish school classes. The pixels of the segmentation map are colored dark blue for background, cyan for class 1, and orange for class 2.

the predictions from the model on an echogram from the $r(f)$ test can be seen in Figure 4.4. From Figure 4.4 we observe that the model's predictions are opposite of the true segmentation mask for several schools. By interchanging the model's predictions between the classes, the predicted segmentation mask would be closer to the true segmentation mask. Interchanging the classes in this way, returns the problem to the combination of shape and relative frequency response that can be found in the shape test. This gives reason to believe that the model bases its predictions on the relative frequency response more than the shape of the schools.

To test if interchanging the relative frequency response between classes affects the performance of the model, the accuracy on this test is compared to the accuracy of the model on the baseline test. If there is a significant difference in accuracy, then interchanging the relative frequency response does affect the model's performance. The results from the $r(f)$ test show that the relative frequency response affect the predictions from the model. The pixel accuracy drops for both classes, as seen in Table 4.1. The drop in accuracy is slightly higher for class 1, than for class 2. The accuracy of the model is listed for each class in Table 4.1. For each class, the accuracy on this test and the accuracy on the baseline test are tested for equal distributions by the Wilcoxon rank-sum test. Both for class 1 and class 2, the tests are significant, ($W = 100$, $P < .001$), and the hypothesis of equal distributions is rejected.

The precision-recall curves for class 1 and 2 of the $r(f)$ test show that the model has the poorest performance on this test. The F_1-scores are 0.143 and 0.251, for class 1 and 2, respectively. The moderately better F_1-score of class 2 align with the results from the pixel accuracy.

Chapter 5

Discussion

A model has been trained and tested on simulated data in order to investigate the effect of two features, relative frequency response and shape, on the model's predictions. The model was fitted to echogram data with two types of fish schools present, and its output is a segmentation map where the model predicts (for each pixel) where the fish schools are present. The schools are simulated such that they differ only in the two features investigated: the relative frequency response and shape. The baseline test has similar data to the training set, and the model performance is high. In the two other tests created, the shape and relative frequency response are interchanged on at a time between the two types of fish schools, in order to investigate their influence on the model's predictions.

Interchanging the shape or the relative frequency response affect the model performance. The drop in model performance is most prominent when the relative frequency response is interchanged, c.f. the $r(f)$ test. This indicates that the relative frequency response is highly weighted during the model's predictions. A possible explanation for why the relative frequency response affect the model's predictions more than shape could be that the relative frequency response affect more pixels in the echogram than what shape does. For the model to detect the shape of the school, it must detect the edge of the schools, i.e., the model only needs the pixels around the circumference of the schools. The relative frequency response of the schools affect the value of all pixels within each school.

The results from the shape test, where the shape of the schools has been altered, show that the model struggles more with predicting schools that were originally spheroids in the simulation and are now ellipsoids, than with schools were the opposite change was made. The schools that were originally spheroids in the simulation was labeled as class 1 in the segmentation maps, while schools that were shaped as ellipsoid was labeled as class 2. In the echograms the spheroidal schools have more backscatter in the depth direction. This means that in the crops the model is trained on, schools belonging to class 1 have a larger circumference than class 2. Following our reasoning above, the shape will then have a larger impact on the model's performance on class 1, as the ratio between circumference and area of the schools is lower, making the shape a more important feature.

The same relationship between the classes can be seen in the $r(f)$ test as well, where the relative frequency responses of the classes are interchanged. The model has a slightly worse performance when predicting class 1, where the relative frequency response has been changed from $r_1(f)$ to $r_2(f)$, than when predicting class 2, where the relative frequency response has been changed from $r_2(f)$ to $r_1(f)$. However, in the calculations of the F_1-score, the threshold used is so low that the F_1-score does not give any real insight to the model. In the case of the accuracy metric, as mentioned in Section 2.2.8, it should not be used alone in the case of imbalanced data. The accuracy of the model on this test could mean that the model is slightly better at predicting class 2, but it is also possible that the model is better at predicting where class 2 is not (the true negatives). Therefore, we can not conclude that the model is better at predicting class 2, than class 1, for this test.

From the baseline performance of the model it is clear that a model can be successfully fitted to simulated data. Simulated data is both cheaper and less time-consuming to acquire, compared to real data. In addition, by using simulated data, the two classes of schools could be created such that they only differ in shape and relative frequency response. This makes it possible to test the model's predictions by changing just one variable at a time. With real data, two fish schools from different fish species will differ in numerous ways. For example, the target strength for all targets in the simulation is based on the estimated target strength for herring. In real data, the target strength will differ between fish species. In addition to the difference in fish characteristics, the data gathered from several trawl surveys may also differ due to extraneous variables, variables that are not intended to be researched. This might be animal sounds or movement, or wind and breaking waves (Simmonds and MacLennan, 2005). The choice of training set and test set is important to create a generalized classification model, and due to extraneous variables, this is a harder choice when data from trawl surveys are used. With simulated data, no variables that are not accounted for will influence the data, making the choice simpler.

Another advantage of the simulated data is the consistency of how the annotations are made. With real data, the process of classifying acoustic backscatter is done by an operator. Brautaset et al. (2020) found both missing annotations and incomplete annotations in the data used to train and test their neural network. The annotations used in this thesis are based on an algorithm that checks the same assumption for all pixels, avoiding inconsistent annotations.

Networks trained on simulated data, can not be directly used on data obtained from surveys, nonetheless they can be useful. Transfer learning is the method of applying a pre-trained neural network on new data (usually from a related domain). The pre-trained network on simulated data can be fine-tuned to survey data, by retraining the higher layers of the network where the features are more case-specific. The lower layers of the network are often useful as they are, because they often detect less abstract features (Azizpour et al., 2015). A network which is pre-trained on simulated data can therefore be a good starting point when training a network on real data.

The relative frequency responses used in this thesis are obtained from calculations done by Holmin et al. (2012), and the report by Fernandes et al. (2006). The function $r_1(f)$ is therefore based on calculations from herring surveys, and $r_2(f)$ is based on data from Atlantic mackerel surveys. These have an increasing difference for higher frequencies. It is possible that the model would be more dependent on the shape of the schools if the relative frequency response was harder to separate. With the pipeline created for this thesis, this is possible to investigate by replacing the relative frequency response functions before training and testing the network again.

Deep neural networks have been shown to perform well on a range of classification problems, but they are often criticized for being difficult to interpret. This thesis sheds some light upon how the model suggested by Brautaset et al. (2020) works, when it is classifying echograms. Other techniques could also be investigated to understand more of the network, e.g. Class Activation Maps (Zhou et al., 2016). These maps highlight the regions which cause the maximum activation of a layer in the network. These will indicate what region in the image, or in this case echogram, which is causing the model to predict the class it is predicting. However, we have instead focused on why the model predicts as it does, and the tests done answer this question for the specific case where the model has to choose between two features; shape and relative frequency response.

In this thesis we have created a pipeline for empirical testing of neural networks on acoustic data. A convolutional neural network has been successfully fitted to simulated acoustic data, and the network achieves a high performance when classifying the acoustic backscatter. The model is tested on several data sets to investigate whether it uses an energetic characteristic (the relative frequency response), or a morphological characteristic (the shape), of the fish schools to segment the echograms. Based on these simulations we conclude that the relative frequency response has a greater impact on the model's predictions than the shape of the fish schools.

Bibliography

Azizpour, H., Razavian, A., Sullivan, J., Maki, A., and Carlsson, S. (Nov. 2015). "Factors of Transferability for a Generic ConvNet Representation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38. DOI: 10.1109/TPAMI.2015.2500224.

Brautaset, O., Waldeland, A. U., Johnsen, E., Malde, K., Eikvil, L., Salberg, A.-B., and Handegard, N. O. (Jan. 2020). "Acoustic classification in multifrequency echosounder data using deep convolutional neural networks". In: *ICES Journal of Marine Science*. ISSN: 1054-3139. DOI: 10.1093/icesjms/fsz235.

Devore, J. and Berk, K. (2011). *Modern Mathematical Statistics with Applications*. eng. Springer Texts in Statistics. New York, NY: Springer New York. ISBN: 978-1-4614-0390-6.

Fallon, N. G., Fielding, S., and Fernandes, P. G. (Apr. 2016). "Classification of Southern Ocean krill and icefish echoes using random forests". In: *ICES Journal of Marine Science* 73.8, pp. 1998–2008. ISSN: 1054-3139. DOI: 10.1093/icesjms/fsw057.

Fernandes, P., R.J., K., Lebourges-Dhaussy, A., Masse, J., Iglesias, M., and Ona, E. (2006). *The SIMFAMI project: Species identification methods from acoustic multifrequency information. Final report to the EC Number Q5RS-2001-02054.*

Gastauer, S., Scoulding, B. C., and Parsons, M. (2017). "An Unsupervised Acoustic Description of Fish Schools and the Seabed in Three Fishing Regions Within the Northern Demersal Scalefish Fishery (NDSF, Western Australia)". In: *Acoustics Australia* 45, pp. 363–380.

Glorot, X., Bordes, A., and Bengio, Y. (Apr. 2011). "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, pp. 315–323.

Gonzalez, R. C. and Woods, R. E. (2018). *Digital Image Processing*. New York, NY: Pearson.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Haralabous, J. and Georgakarakos, S. (Apr. 1996). “Artificial neural networks as a tool for species identification of fish schools”. In: *ICES Journal of Marine Science* 53.2, pp. 173–180. ISSN: 1054-3139. DOI: `10.1006/jmsc.1996.0019`.

He, H. and Garcia, E. A. (Sept. 2009). “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9, pp. 1263–1284. ISSN: 2326-3865. DOI: `10.1109/TKDE.2008.239`.

Hinton, G. E. (1992). “How Neural Networks Learn from Experience”. In: *Scientific American* 267.3, pp. 144–151. ISSN: 00368733, 19467087.

Holmin, A. J., Handegard, N. O., Korneliussen, R. J., and Tjøstheim, D. (2012). “Simulations of multi-beam sonar echos from schooling individual fish in a quiet environment”. In: *The Journal of the Acoustical Society of America* 132.6, pp. 3720–3734. DOI: `10.1121/1.4763981`.

ICES (Jan. 2008). *Report of the Planning Group for Herring Surveys (PGHERS)*, 256 pp.

Ioffe, S. and Szegedy, C. (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML’15. Lille, France: JMLR.org, pp. 448–456.

Kinsler, L. E., Frey, A. R., and Mayer, W. G. (2000). *Fundamentals of Acoustics*. Fourth Edition. New York: John Wiley and Sons, Inc. ISBN: 0471847895.

Kloser, R. J., Ryan, T., Sakov, P., Williams, A., and Koslow, J. A. (2002). “Species identification in deep water using multiple acoustic frequencies”. In: *Canadian Journal of Fisheries and Aquatic Sciences* 59.6, pp. 1065–1077. DOI: `10.1139/f02-076`.

Korneliussen, R. J. and Ona, E. (Mar. 2002). “An operational system for processing and visualizing multi-frequency acoustic data”. In: *ICES Journal of Marine Science* 59.2, pp. 293–313. ISSN: 1054-3139. DOI: `10.1006/jmsc.2001.1168`.

– (Jan. 2003). “Synthetic echograms generated from the relative frequency response”. In: *ICES Journal of Marine Science* 60.3, pp. 636–640. ISSN: 1054-3139. DOI: `10.1016/S1054-3139(03)00035-3`.

Korneliussen, R. J., Heggelund, Y., Eliassen, I. K., and Johansen, G. O. (May 2009). “Acoustic species identification of schooling fish”. In: *ICES Journal of Marine Science* 66.6, pp. 1111–1118. ISSN: 1054-3139. DOI: `10.1093/icesjms/fsp119`.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., pp. 1097–1105.

Lecun, Y., Bengio, Y., and Hinton, G. (2015). “Deep learning”. In: *Nature* 521, pp. 436–444. DOI: `https://doi.org/10.1038/nature14539`.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. Berlin, Heidelberg: Springer-Verlag, pp. 9–50. ISBN: 3540653112.

Long, J., Shelhamer, E., and Darrell, T. (2015). “Fully convolutional networks for semantic segmentation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440.

MacKay, D. J. (2003). *Information Theory, Inference and Learning Algorithms.* Cambridge University Press. ISBN: 9780521642989.

Murphy, K. P. (1993). *Machine Learning: A probabilistic Perspective.* Cambridge, Massachusetts: The MIT Press.

Polyak, B. (1964). “Some methods of speeding up the convergence of iteration methods”. In: *USSR Computational Mathematics and Mathematical Physics* 4.5, pp. 1–17. ISSN: 0041-5553. DOI: `https://doi.org/10.1016/0041-5553(64)90137-5`.

Ronneberger, O., Fischer, P., and Brox, T. (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.* Ed. by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Cham: Springer International Publishing, pp. 234–241.

Simmonds and MacLennan (2005). *Fisheries Acoustics: Theory and Practice.* 2nd ed. Fish and Aquatic Resources Series. Blackwell Science Ltd. ISBN: 978-0-632-05994-2.

Weill, A., Scalabrin, C., and Diner, N. (1993). “MOVIES-B: an acoustic detection description software. Application to shoal species’ classification”. In: *Aquat. Living Resour.* 6.3, pp. 255–267. DOI: `10.1051/alr:1993026`.

Zeiler, M. D. and Fergus, R. (2014). “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV 2014.* Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Cham: Springer International Publishing, pp. 818–833.

Zhou, B., Khosla, A., Lapedriza, À., Oliva, A., and Torralba, A. (2016). “Learning Deep Features for Discriminative Localization”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2921–2929.