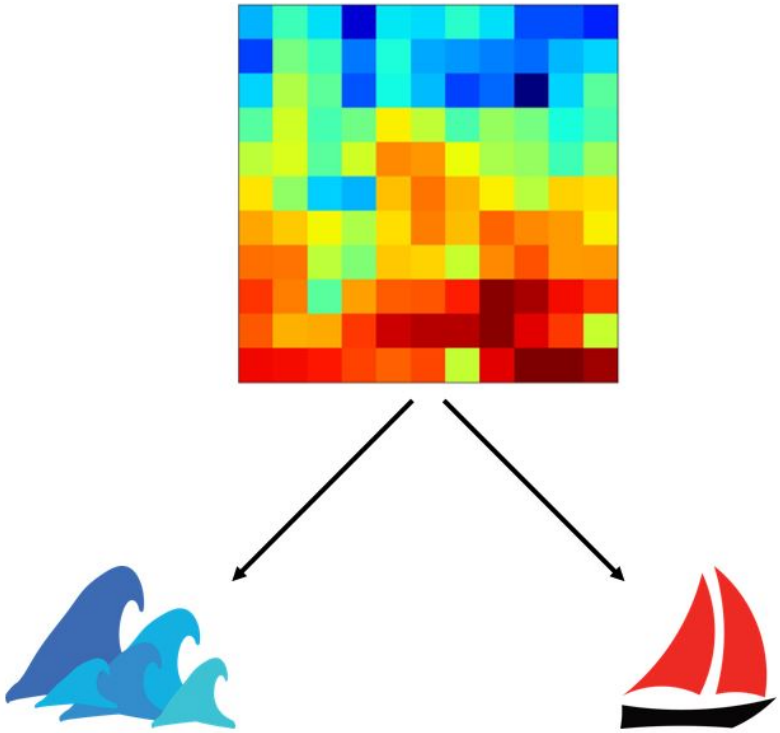
CHALMERS
UNIVERSITY OF TECHNOLOGY



Machine Learning for Categorization of Small Boats and Sea Clutter

Radar Data Processing, Design and Training of Convolutional Neural Network Models

Master's thesis in Systems, Control and Mechatronics

LUDWIG ADOLFSSON

MATTIAS RAHM

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:EX057

Machine Learning for Categorization of Small Boats and Sea Clutter

Radar Data Processing, Design and Training of Convolutional Neural Network Models

LUDWIG ADOLFSSON
MATTIAS RAHM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Machine Learning for Categorization of Small Boats and Sea Clutter
Radar Data Processing, Design and Training of Convolutional Neural Network Models
LUDWIG ADOLFSSON
MATTIAS RAHM

Supervisor:	ANDERS LARSSON	System Design Sensor, Saab AB
	PATRIK DAMMERT	Concepts and Functions, Saab AB
	ARTUR CHODOROWSKI	Department of Electrical Engineering Chalmers University of Technology
Examiner:	FREDRIK KAHL	Department of Electrical Engineering Chalmers University of Technology

Master's Thesis 2018:EX057
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Example patch from radar data with classification categories clutter [1] and boat [2].

Typeset in LaTeX
Gothenburg, Sweden 2018

Machine Learning for Categorization of Small Boats and Sea Clutter
Radar Data Processing, Design and Training of Convolutional Neural Network Models
LUDWIG ADOLFSSON
MATTIAS RAHM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

When using an airborne radar system to search for boats at long distances, a big challenge is to distinguish the echo of small boats from the echo of sea clutter. In a military defence system it is of great importance to be able to monitor all different kind of threats. The desire to constantly improve the efficiency in current systems include monitoring of smaller objects, where the machine learning approach Convolutional Neural Network (CNN) may be one way to streamline categorization of small vessels and sea clutter. The objective of this study is therefor to evaluate the usage of CNN for this categorization problem.

This project has been performed at the System Design Sensor unit at the Airborne Surveillance Systems of Saab AB. To achieve the objective, radar data provided by Saab are used. Training data consist of samples of sea clutter and targets. The clutter samples are extracted from the available radar data and the targets in this study are created with the help of a developed algorithm that generates synthetic small boat like targets. The synthetic modelling is performed in order to be able to generate a large amount of target samples. The classifier used to categorize the clutter from the targets is modelled as a CNN where hyperparameter tuning is performed as well as evaluation of architectural parameters.

The results show that a CNN can be used to categorize small boats and sea clutter with a cross-validated classification accuracy above 90%. It is concluded that the structure of the training data have a greater impact on the performance than the network parameters. The distribution of the amplitudes of the synthetic targets is similar to that of the sea clutter's. This is good since it reflects the reality where the small boats are similarly looking as the clutter. Further investigation of the target model is however recommended. It is also recommended to evaluate the performance of the classifier on real targets.

Keywords: Machine Learning, Image Processing, Pulse Radar, Convolutional Neural Network, Deep Learning, Sea Clutter

Acknowledgements

Firstly, we would like to give special thanks to our supervisor at Saab AB, Anders Larsson, who has been very encouraging and supportive throughout the entire project. We would also like to thank our other supervisor at Saab AB, Patrik Dammert, who has contributed with valuable insight along the project, and especially with the synthetic target modelling. Furthermore, Saab AB has been a good place to work where we have been well received and have been included in the team. We have been pleased to be a part of Saab AB during this spring.

Finally, we thank our examiner Fredrik Kahl and supervisor Artur Chodorowski at Chalmers University of Technology for taking on this project. The project has been very interesting and a good way of finishing the master's programme Systems, Control & Mechatronics.

Ludwig Adolfsson & Mattias Rahm, Gothenburg, May 2018

Contents

List of Abbreviations

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
FC	Fully Connected
FFT	Fast Fourier Transform
FN	False Negatives
FP	False Positives
GD	Gradient Descent
PRF	Pulse Repetition Frequency
PRI	Pulse Repetition Interval
Radar	Radio Detection And Ranging
ReLU	Rectifier Linear Unit
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
TN	True Negatives
TP	True Positives

List of Figures

List of Tables

1

Introduction

Airborne naval surveillance is used to search for boats at long distances, where ground-based radar systems are limited to detect objects by the horizon. A big challenge is to distinguish the echo of small boats from radar echoes of sea reflections and waves, known as sea clutter. Sea clutter can be relatively strong and it is hard to statistically model its behaviour since the sea is unpredictable. In the work of this project, real flight test data with sea clutter combined with simulated objects are used.

In the machine learning area, considerable progress has been achieved in recent years. This has been made both at basic level, such as deep-learning, but also at low levels with better principles such as improved learning with a Rectifier Linear Unit (ReLU) as activation function in neurons. Dropout is also commonly used as a regularization technique in order to reduce the risk of overfitting when training neural networks. The purpose of this project is to evaluate how well these, amongst others, principles can be used to classify objects at sea from sea clutter.

This project has been performed at the System Design Sensor unit at the Airborne Surveillance Systems of Saab AB. The Surveillance business area provides effective monitoring and decision support solutions, and systems for detecting and protecting against different types of threats. System Design Sensor is responsible for system development of sensors, such as radars. The responsibility includes system design of signal and data processing functions, advanced simulation and more.

1.1 Objective

The objective of the project is to perform a study of whether the machine learning approach of using a Convolutional Neural Network (CNN) can be used when categorizing small boats from sea clutter in radar data. To achieve the objective, preprocessing of radar data is performed and different network architectures with varied hyperparameter settings are created, tested and evaluated. The best performing network is used when evaluating data parameters such as the amount of input training samples and the size of each input sample.

1.2 Purpose and Aims

The main purpose of this project is to investigate the possibilities of using a CNN for classification of small boats and sea clutter. Together with the main purpose, the following questions are used to clarify and evaluate the aims of the project

- A neural network can be structured in different ways, e.g. different amount of layers or neurons in each layer. Which network architecture solves the problem best?
- There are parameters such as learning rate and batch size, which are experimented with when training the network. Which parameter choice is best suited?
- Will increased amount of training data improve the results, and is there a limit of the amount where the results stagnate?
- Do different noise levels, based on weather condition, have any impact on the classification performance?

1.3 Scope and Boundaries

This project is focusing on implementing an offline algorithm that uses radar data from previous recorded flight tests, generating training data and training a neural network for binary classification, i.e. sea clutter or boat. A possible implementation in a real time system is outside the scope of this project as well as considering time sequences of the radar data. When generating training data the boats, also known as targets, are synthetic in order to simplify the data generation. The targets are also placed in different sea conditions.

1.4 Outline

In Chapter 2 relevant background theory is presented in order to incorporate a better understanding of relevant techniques and terms used throughout the report. Chapter 3 presents the methodology used when working out the best way of solving the objective of this project, including processing of radar data, creating training data and evaluation of network as well as data parameters. In Chapter 4 tables of chosen parameters to test, together with results of different combinations, are presented. This chapter also contain discussion regarding the results produced. In Chapter 5, the conclusion of this project is presented together with suggestions for future work.

2
Theory

In order to understand the methodology of this report, basic fundamentals of related theory are presented in this chapter. The data set which is used to perform the machine learning analysis on is obtained from raw radar data, which is being preprocessed in different manners as described in Chapter 3. To get a better understanding of how this data is structured, knowledge of the methodology behind radar is presented in Section 2.1. Since the objective of this study is to categorize boats and sea clutter, an introduction to the concept of sea clutter is described in Section 2.2. The theory behind machine learning is presented in Section 2.3 with focus on describing the machine learning approach used in this study.

2.1 Radar

Radar is short for Radio Detection And Ranging. The abbreviation describes what is used, namely radio waves, and what the purpose of radar is, that is to detect objects and determine the range and direction of these objects [3]. The arrangement of a basic radar, with one antenna being used for both transmitting and receiving signals, is shown in Figure 2.1. The function of the transmitter is to send radio waves in a certain direction via the antenna. The back scatter, also known as echo or return, is then picked up by the same antenna and processed by the receiver in a desirable way to be shown on a display. Since the same antenna is used for transmitting and receiving signals, both operations can not be performed simultaneously.

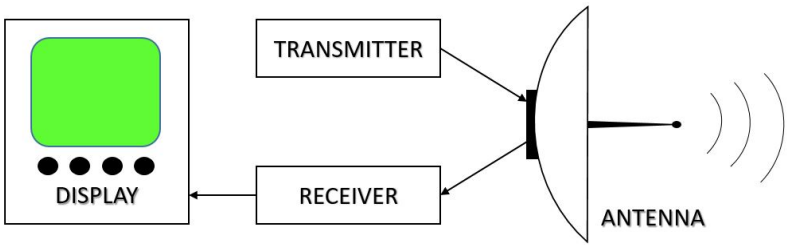


Figure 2.1: Arrangement of basic radar with transmitter, antenna, receiver and display.

In order for the radar to function in a proper way the transmitter does not send radio waves continuously, instead pulses are used where the antenna transmits and receives signals alternately. This is called a pulse radar. The pulses are transmitted in a rate known as the Pulse Repetition Frequency (PRF) where the receiver is

activated in between the pulses allowing listening for echoes. The Pulse Repetition Interval (PRI) is defined as PRI=1/PRF [3].

The aim of the radar is to detect objects by optimizing a search pattern for different types of targets, e.g. objects at long distances or fast moving objects. The PRF is chosen based on the type of the targets in focus. In this study returns fulfilling certain conditions, which are specified in Chapter 3, are considered as sea clutter. To be able to distinguish different objects from each other and to detect objects at long distances the radar antenna uses a concentrated narrow beam which is swept over a frame. The frame can be visualized as a circular sector, as shown in Figure 2.2, where the radar beam moves a certain amount of degrees in a given direction, clockwise or anti-clockwise, until the end of the sector is reached and the scanning starts over. Each position of the beam is called a lobe position and for each lobe position a specified amount of pulses are transmitted and echoes are received [3].

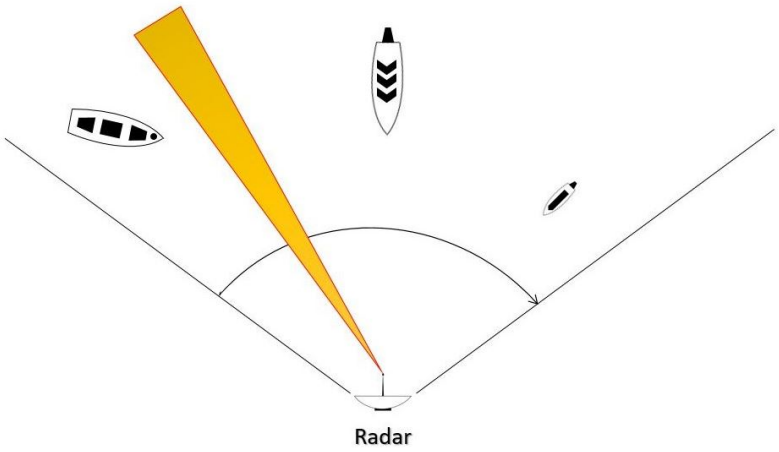


Figure 2.2: Search frame with current lobe position from radar antenna, scanning direction and boats.

For an object to be detected, obstacles of which the radio waves cannot penetrate, must not be apparent. Stimson [3] calls this the line of sight, and continues explaining that even though the object may be fully visible, noise from the receiver output and/or ground clutter may generate a stronger signal than the object. In order to exceed the background noise level the object should be within a certain range from the radar. According to [3] there are a number of aspects deciding at what range the object is detectable. Size of the antenna, reflecting characteristics, wavelength of the radio waves, strength of background noise and power of the transmitted pulse to name a few.

To determine the range between the radar and the object, Stimson [3] gives the following relationship

$$\text{Range} = 0.5 \cdot \text{Round Trip Time} \cdot \text{Speed of Light}$$

where the Round Trip Time is defined as the time between the transmitted pulse

and the received echo. Radio waves move in the Speed of Light, which is equal to 299,792,458 meters per second. The number 0.5 is present since the range to the object is half the range the pulse has traveled when being received by the radar.

According to [3], there exist a trade-off between distinguishing objects close to each other and detecting objects at long distances. In order to separate returns from objects close to each other, the pulse width has to be small enough. On the other hand, a contradiction occurs when detecting objects far away as the energy from the radar has to be increased, demanding a wider pulse width. The trade-off is solved by pulse compression of the received complex signal which makes it possible to transmit wider pulses without the downside of not being able to separate objects from one another in the range domain.

When the range rate of an object varies, a shift in the frequency of the transmitted signal versus the received signal is obtained. The shift is proportional to the range rate and the effect of the shift is called the Doppler effect [3]. This effect allows separation of moving objects from ground clutter. When studying maritime objects the ground clutter is equal to the sea return. An example of this is shown in Figure 2.3.

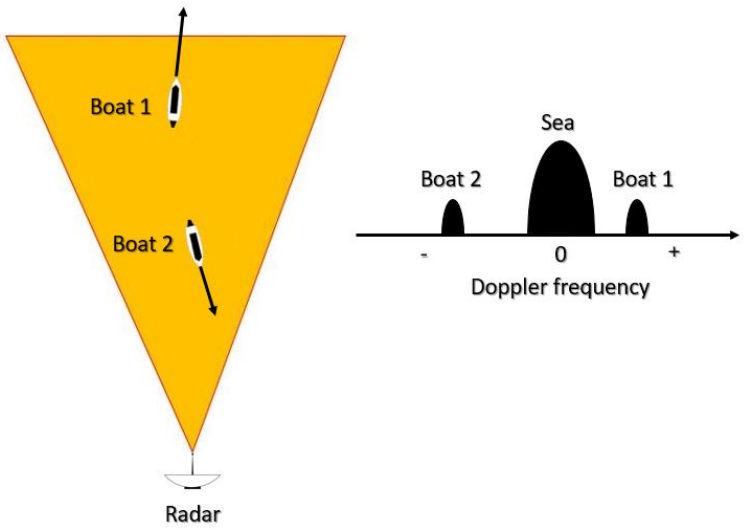


Figure 2.3: Two boats moving in different directions give rise to Doppler effects, which are separable from echoes of the sea.

In order for several objects to be separable in the Doppler domain, a transformation is performed of the returned complex signal. The transformation is made by using the Fast Fourier Transform (FFT) which return a narrow peak based on the frequency and the amplitude of the signal.

The procedure of compressing the radar data in range domain and transforming in Doppler domain is visualized in Figure 2.4, where the complex signals received are shown after being logarithmed. The logarithmic operation is performed in order for

improved visual effect.

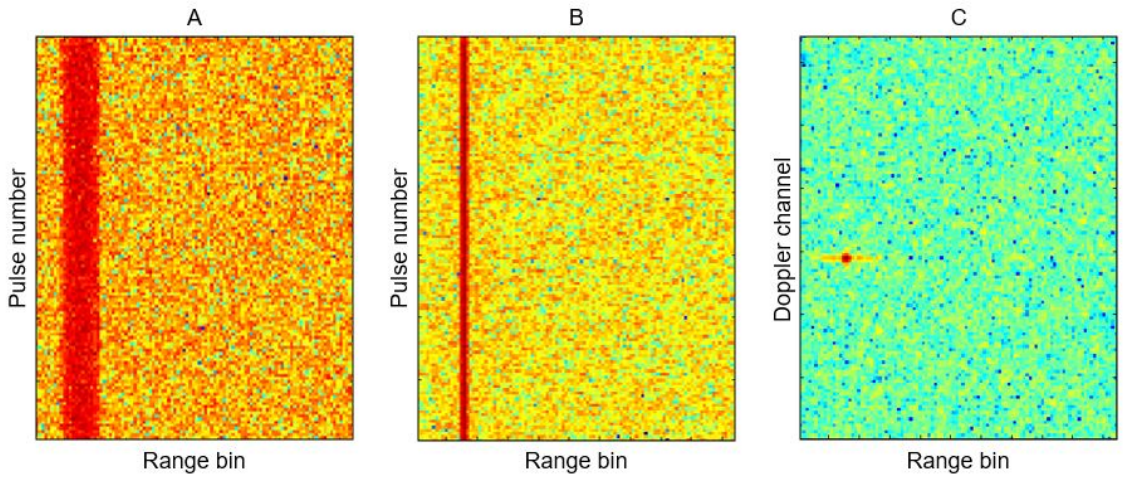


Figure 2.4: Logarithmed plots before and after pulse compression and FFT. A: Example of input data from the radar. B: Data after being pulse compressed. C: Data after pulse compression and FFT.

2.1.1 Airborne Radar

In this study, maritime objects are investigated from airborne radar. In Figure 2.5 the placement of the radar used when collecting the radar data is shown together with the frame over which the beam is swept. As can be seen, the radar used to collect the raw data for this study is placed on top of the aircraft scanning surroundings in sector shaped frames.

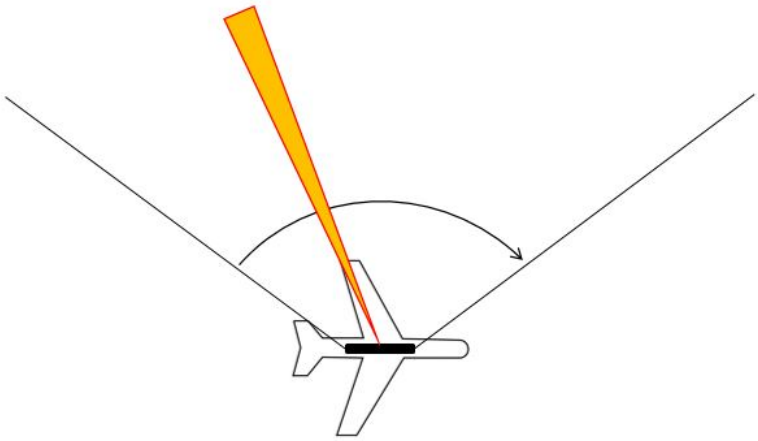


Figure 2.5: Aircraft with roof mounted radar seen from above with radar frame section, current lobe position and scanning direction.

An example plot of how the collected data, after pulse compression and FFT, may look like is shown in Figure 2.6. This data is collected when monitoring sea. The

colors in the plot are related to the intensity of the returns. The more red a pixel is the larger the amplitude of the echo is, which means that a blue pixel generate a small echo whereas a dark red pixel generate a large echo. In the middle of the Doppler domain there is a band of reddish pixels which represent the sea return. The variations in the Doppler domain for this band are due to different velocities on the sea surface. The greenish bands above and under the sea band represent noise. There are a few reddish pixels in the noise bands, which have a significantly larger amplitude than their surroundings. These pixels are objects which have another velocity than the sea, and may thus be targets such as boats. A boat which generate an echo with the same Doppler frequency as the sea return will end up in the sea band, and small boats within this region are very hard to detect since their amplitude will not exceed the amplitude of the sea.

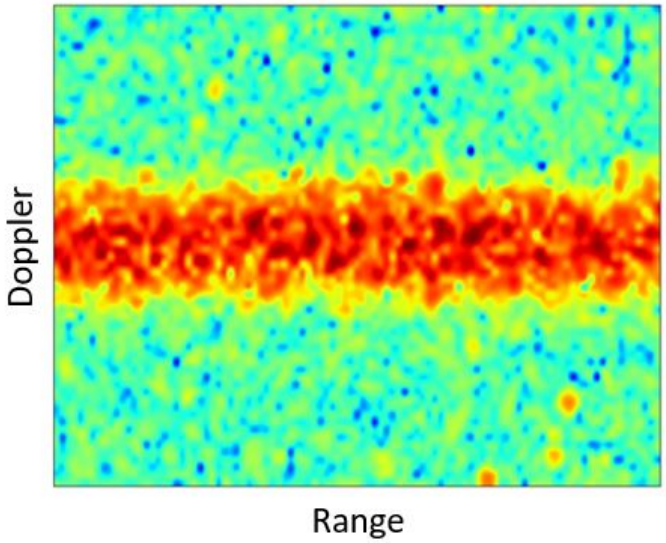


Figure 2.6: Example plot from radar collecting flight, after pulse compression and FFT.

2.2 Sea Clutter

When transmitting radio waves towards maritime objects most of the returns are generated by echoes from the sea. In the case of airborne naval surveillance the back scatter from sea clutter is unwanted, since the objective is to detect maritime objects rather than observe the motions of the sea. The hard task when coping with sea clutter is thus that the sea, and especially breaking waves, generate amplitudes and velocities that are similar to that of small vessels. According to Ward et. al. [4] airborne radars, with low grazing angle at the sea surface, have a harder task to distinguish targets from sea clutter. Apart from viewing geometry, there are other aspects such as weather conditions and radar parameters making separation of targets and sea clutter more or less difficult. Weather conditions may refer to the state of the sea, the wind velocity or wave height which is unpredictable thus hard to model statistically. Examples of radar parameters are range resolution and

transmission frequency.

As explained in Section 2.1, the Doppler effect may be used to separate targets from sea clutter returns. Ward et. al. [4] also mention this approach and continue explaining how the radar needs to be able to distinguish targets with Doppler shift close to the sea clutter's Doppler spectrum. Targets matching this criteria has a low radial velocity and are thereby harder to separate from the large return of the sea.

2.3 Machine Learning

Machine learning is the concept of using experience from input data, often large data sets, in order to obtain knowledge by finding some sort of pattern. One advantage of using machine learning over conventional methods is its ability to generalize to new unseen data, called adaptivity. Shalev-Shwartz and Ben-David [5] describe how machine learning can be used in situations where it is hard for a human to script an explicit enough program to work for all possible outcomes. An example of this is driving a car, where continuous learning is essential in order for the program to perform well. Another example involves large and complex data sets of images where the human eye may have difficulties of seeing underlying features whereas the computer is allowed to scan through the data in a faster and more structured way.

The work of outputting a prediction rule, based on input, is called classifying. Shalev-Shwartz and Ben-David [5] highlight two types of learning a classifier

- Supervised Learning
- Unsupervised Learning

Supervised learning is performed by feeding a classifier with information, called labels, together with the training examples. Each label corresponds to an example of the input data. By using labels it is demanded of the user to label data accordingly in order for the classifier to perform in a proper way. The labeling process thus requires knowledge of the input data, and may be time consuming when handling large data sets if the labels are to be set manually.

Unsupervised learning does not demand labeling. The methodology of this approach is rather to find anomalies in the data and sort input data with similar features into clusters.

When using the supervised learning approach, the task is to use a domain set, $X = (x_1, x_2, ..., x_p)$, containing samples that are to be sorted in accordance with the categories spanning the label set, Y. The amount of samples in the domain set is here denoted as p. In order to perform this task on new unseen data, a training set, S, with m amount of samples is used in the form $S = ((x_1, y_1), (x_2, y_2), ...(x_m, y_m))$, where y_i is one of the possible labels from Y and $m < p$. The classifying problem can be viewed as an optimization problem where the objective is to minimize a loss

function. There are multiple choices of loss functions to be used depending on the nature of the problem, e.g. binary and multiclass classification problems. The better the prediction of the classifier the lower the loss. The measure of the performance of a classifier may be evaluated by examining what percentage of correctness the prediction achieve, namely the accuracy, as well as monitoring the loss [5].

The accuracy is proper to use as measurement as long as the data set is balanced, which means that there are equal amount of the different categories in the training set S. If this is not the case, i.e. S is unbalanced, the classifier could obtain a misleading high accuracy by always predicting the most common of the categories. Even though S is balanced and the accuracy is high when training it does not necessarily mean that the result is good, which is problematic. This problem that may arise is called overfitting, which Shalev-Shwartz and Ben-David [5] describe as the case where the training has led to the classifier being not generalizing enough. An example of overfitting happens when training a complex classifier with too little data, leading to high accuracy when training but poor accuracy when using the classifier on new unseen data. In order to detect the problem of imbalance or overfitting, a validation set, $V = ((x_{m+1}, y_{m+1}), ..., (x_n, y_n))$, may be used. Here, $n < p$ in order to leave some samples for testing. The accuracy and the loss of the validation set thus work as a check of whether the classifier is exposed to overfitting or not. An example of overfitting is shown in Figure 2.7. As can be seen the test accuracy increases over time, while the validation accuracy stagnates. As for the loss, the training loss keeps decreasing, while the validation loss starts increasing at around epoch 10.

Finally a test set, $T = ((x_{n+1}, y_{n+1}), ..., (x_p, y_p))$, consisting of unseen data, often is predicted to obtain the final performance of the classifier [5]. The test set is to be used only once in order for the classifier not to memorize any of the test objects, which may generate misleading results.

2.3.1 Stochastic Gradient Descent

The objective of learning is to minimize a loss function. Gradient Descent (GD) is a way of minimizing the loss function by taking a step in the opposite direction of the gradient of the loss function. This is performed after one pass over the entire training set. According to [6], when handling large data sets it can be very time consuming and require a lot of memory capacity to use the whole training set when computing the next parameter update towards a local optima. Stochastic Gradient Descent (SGD) is a way of handling this complexity by using stochastically chosen samples, called mini batches, from the training set and taking a step in the negative direction of the gradient for the loss function based on the mini batch instead of the entire training set [5]. The high computational cost of using the full training set, as in regular GD, can be overcome with SGD and still lead to fast convergence which is why it is often used in neural networks where the data sets often are very large. When using SGD it is important to shuffle the data before training in order to prevent biasing of the gradient, which may lead to bad convergence [6].

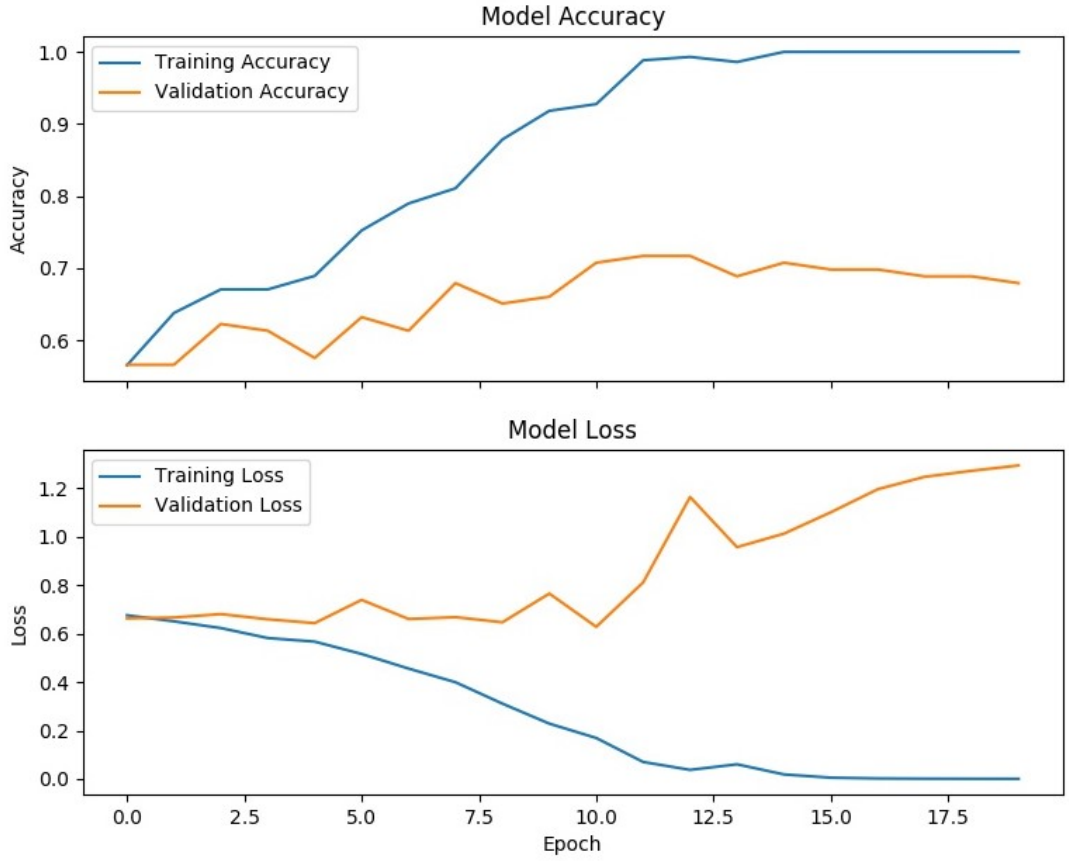


Figure 2.7: Example of overfitting for a model. At approximately 10 epochs, the validation accuracy stagnates while the validation loss increases, i.e. the model is overfitting.

In [7], the update rule for SGD is written as

$$\theta^{k+1} = \theta^k - \mu \nabla L_i(\theta^k)$$

where θ is a parameter to update, μ is the learning rate and $\nabla L_i(\theta^k)$ is the gradient of the loss function in the i^{th} mini batch. To obtain a feeling of how learning may differ with different parameter settings, examples in Figure 2.8 shows noisy 2D measurements of parameter θ (blue dots) together with level curves of the loss function (black circles), steps towards optima (red lines) and the optima (purple square). If μ is set to a fixed and relatively large value throughout the learning, a risk of overshooting occurs when reaching close to the optima, as shown in (A) in Figure 2.8. In order to reduce overshooting, and converge better, it is common to decrease μ the closer we get to the optima, which is called learning rate decay, and is shown in (B) in Figure 2.8. SGD is often combined with momentum, which decides how much impact previous gradients have on the update. This may reduce the risk of oscillating when finding the way towards the optima. Momentum may thus decrease the time it takes to converge. The update rule, together with the estimate g^k of the gradient of the loss, is in [7] written as

$$\theta^{k+1} = \theta^k - \mu g^k$$

$$g^k = \gamma g^{k-1} + (1-\gamma)\nabla L_i(\theta^k)$$

where $\gamma \in (0, 1]$ is the momentum factor. The result of learning with momentum and learning rate decay is shown in (C) in Figure 2.8.

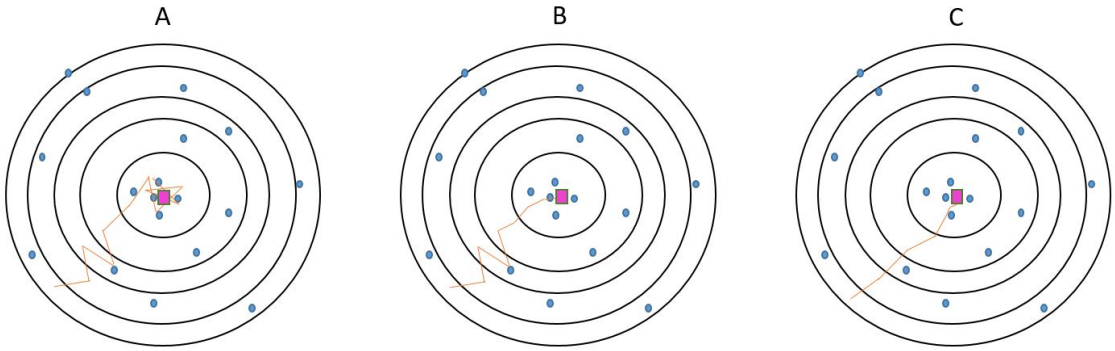


Figure 2.8: A: Shows how it is hard to reach the optima when using a constant, quite large, learning rate. B: Shows how learning rate decay helps reaching the optima, without overshooting. C: Shows a smoother way towards the optima, with the help of using momentum.

2.3.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are based on the structure of the human brain, where neurons are connected in a structure to solve highly complex problems by communicating via dendrites [5]. In Figure 2.9 a basic model of a neuron is shown. It can be seen that the neuron receives inputs (x_1, x_2, x_3) which are multiplied with weights (w_1, w_2, w_3), defining what to pay extra attention to, and then summed. The weights may be initialized in different ways. Two common choices are either zeros or small random numbers, where the latter is preferred to avoid identical derivatives and instead make different neurons diverge in order to allow different structures [7]. In addition to the inputs, a bias weight is added as w_0. The bias term is added to be able to shift the function, since the optimal model of the problem may include a shift from the origin. An activation function, σ, is then used to represent the activation of a biological neuron [7].

In order for the brain to solve complex problems, a large amount of neurons are used. In ANNs the structure is built up by using layers with nodes, representing the neurons, and edges, representing the dendrites. The edges connect the nodes to form a network. The layers between the input and the output layers are called hidden layers [5]. The input to each node is the weighted output from previous layer of nodes, which may be Fully Connected (FC) layers. In Figure 2.10 an example of an FC feedforward network is shown with inputs x_1, x_2 and x_3, a hidden layer with neurons, connecting edges and an output layer. The output layer may, in a multiclass classifying problem, be a softmax where each class is given a probability. The sum of all probabilities is equal to one and the predicted label is set to be the same as the class with the highest probability. All nodes, except the ones in

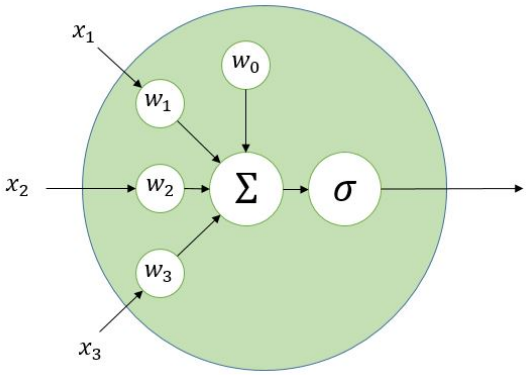


Figure 2.9: Schematic figure of a neuron. The neuron is fed with inputs x_1, x_2, x_3 with corresponding weights w_1, w_2, w_3. The weighted inputs are summed with a bias weight w_0 and passed through an activation function σ.

the input layer which simply feed forward values from the input vector, in Figure 2.10 are built up by the same structure as the neuron in Figure 2.9. When training a network, a structure is predefined and the learning proceeds as the weights are continuously updated in order to minimize the loss function. There are a number of optimizers to use when minimizing the loss function, where many of them are different variants of the SGD implementation.

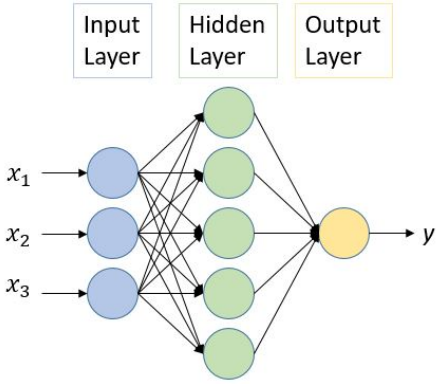


Figure 2.10: Structure of an example network with inputs (x_1, x_2, x_3), nodes, edges and output y.

In [5], a number of features of the network are defined

- Size The number of neurons in the network
- Depth The number of layers in the network, not including the input layer
- Width The number of neurons in the widest layer

For the example network in Figure 2.10 the size is equal to 9, the depth is equal to 2 and the width is equal to 5.

A way to reduce the risk of overfitting when using a neural network is to use a regularization technique called dropout. By adding this feature a given percentage of a

layer's neurons are removed at each iteration of SGD. By doing this it is possible to recognize the object in multiple ways [7]. When removing the neurons, connections are removed as well which can be seen in Figure 2.11.

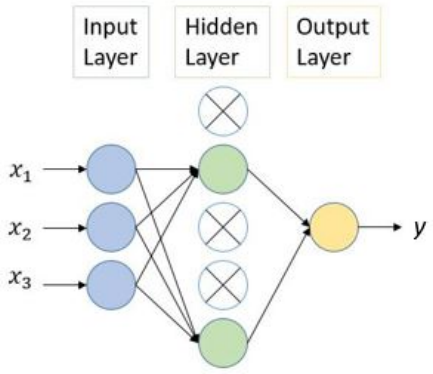


Figure 2.11: Structure of an example network. Three neurons are being removed with dropout.

Another regularization method that may give positive results is to set constraints on the weights. One common way is to use a max norm constraint, which will scale the weight matrix so that the norm does not exceed a certain value [8].

2.3.3 Convolutional Neural Network

When it comes to handling images as inputs, the regular neural networks with Fully Connected (FC) layers are not very effective [8]. The number of weights increases rapidly when the number of neurons and layers increase. As an example, an input image of size 64x64x3 (height, width, channels) will produce 64x64x3=12288 weights for a single fully connected neuron in the first layer. This makes the regular networks computationally demanding to train and with this many parameters it will also be prone to overfitting.

One way to overcome this problem is to use what is called a Convolutional Neural Network (CNN). These networks are similar to regular neural networks in many ways, for example they consist of neurons that have learnable weights and use a loss function to compute the error, but are developed especially for the task of handling images. CNNs make use of some properties that make it possible to reduce the number of parameters considerably compared to a regular fully connected network [8].

The most common layers used in a CNN are described below.

Convolutional layer
The convolutional layer is the main feature of a CNN. Instead of taking a vector as input, it takes an image with height, width and depth. It uses a set of filters that slide over the image, with a certain step size called stride. At each step it produces

an output value. Each filter consists of a set of learnable weights and has a height, width and depth. The depth of each filter is the same as the depth of the input, while the height and width, also known as kernel size, and stride can take different values. At each step the dot product between the filter's weights and the region that the neuron is connected to in the input is calculated. This operation can be seen in Figure 2.12. The spatial size (height and width) of the output volume from a convolutional layer is decided by the filter's kernel size and stride, and the depth of the output is equal to the number of filters used.

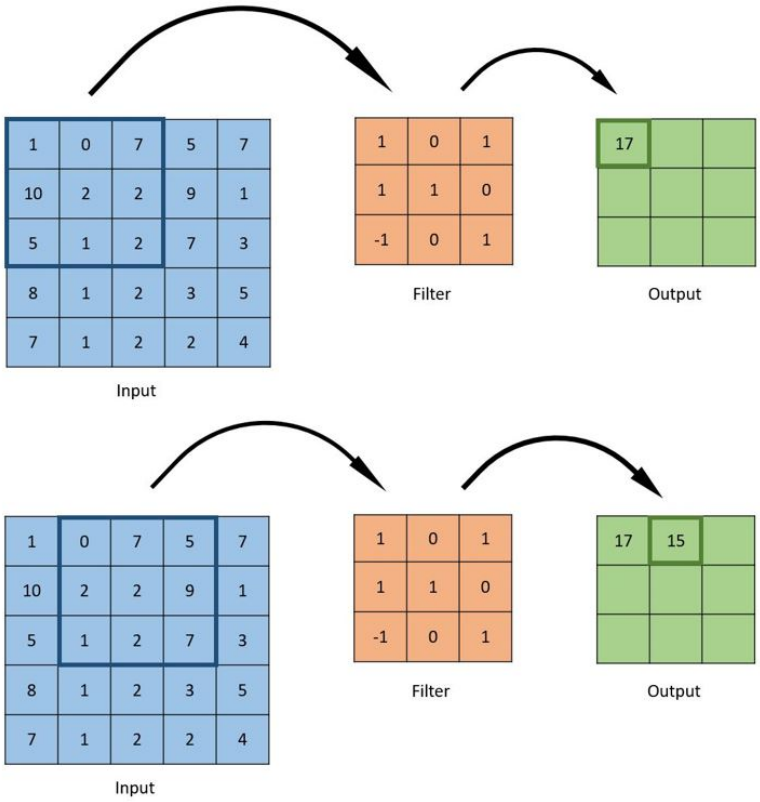


Figure 2.12: The convolutional operation for a single channel input using a filter of size 3x3 and stride 1.

Since each neuron in a convolutional layer only is connected to a small region in the input, how small is decided by the size of the filters, the number of weights are kept at a manageable level. For example, for an input image of size 64x64x3, if filters with kernel size 3x3 are used, the number of weights for each neuron will be 3x3x3=27.

The convolutional layer decreases the spatial size of the input. This can however be controlled by zero-padding the border of the input as seen in Figure 2.13. The most common way is to zero-pad so that the spatial size of the output will be the same as for the input. In that way the information at the borders is preserved [8].

Activation layer

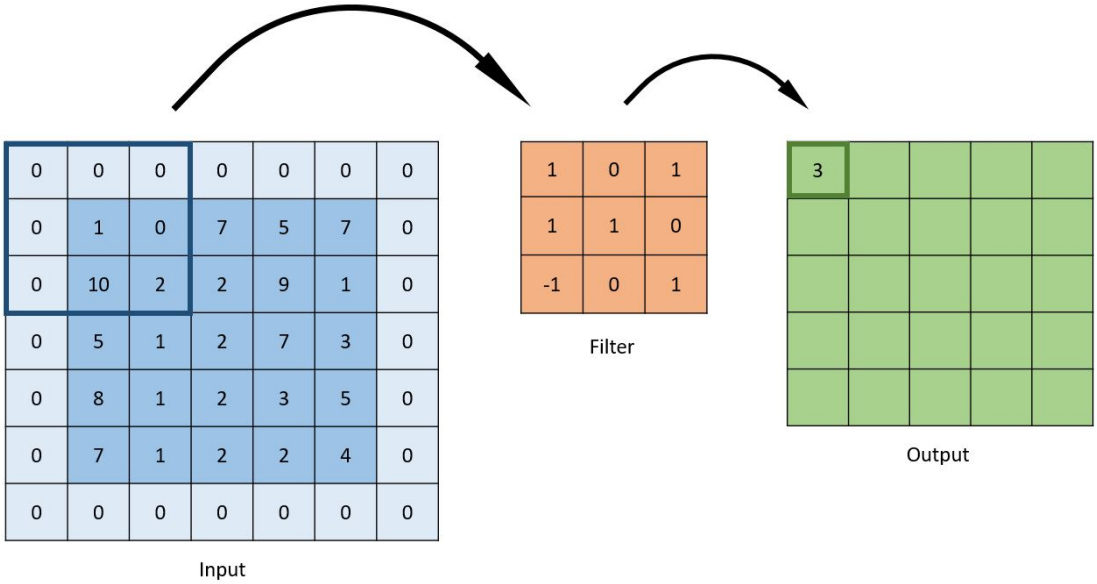


Figure 2.13: The convolutional operation for a single channel input using a filter of size 3x3 and stride 1. Here the input is zero padded and the spatial size is preserved.

The output from a convolutional layer is sent through a nonlinear activation function which is crucial for the networks ability to map all different kind of functions. Without this non-linearity, the network would work as a linear transformation from input to output and lose a lot of its power. The activation function is applied elementwise and will not affect the size of the input. In recent years, the ReLU function has become one of the most popular activation function in CNNs, which simply is a max(0,x) function that sets all negative values to zero.

Pooling layer

After a convolutional layer, it is common to insert a pooling layer which function is to reduce the spatial size of the output in order to decrease the amount of parameters in the network, see Figure 2.14. The pooling layer works in a similar way as the convolutional layer, where a window slides over the input with a certain stride and for each step produces an output value. The most common type of pooling is max pooling which, as the name indicates, outputs the maximum value at each step.

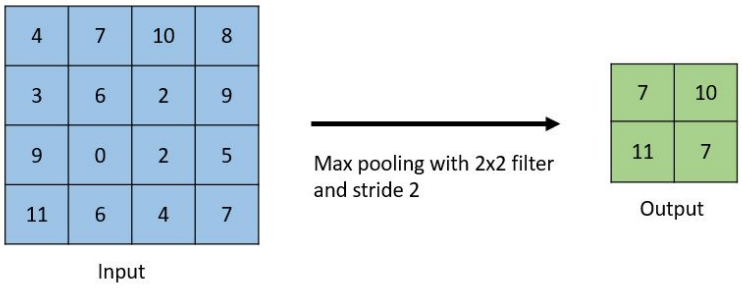


Figure 2.14: The result of a max pooling operation with filter size 2x2 and stride 2.

Fully Connected layer
A CNN often ends with one or more fully connected layers. These are the same as in a regular neural network described in Section 2.3.2. In the last stages of a CNN the input is usually relatively small and hence the fully connected layers will not produce too many parameters.

In Figure 2.15 an example of a CNN for a classification task with five categories can be seen. The input is an image of size 64x64x3 and the output may be the probabilities of the image belonging to the different categories.

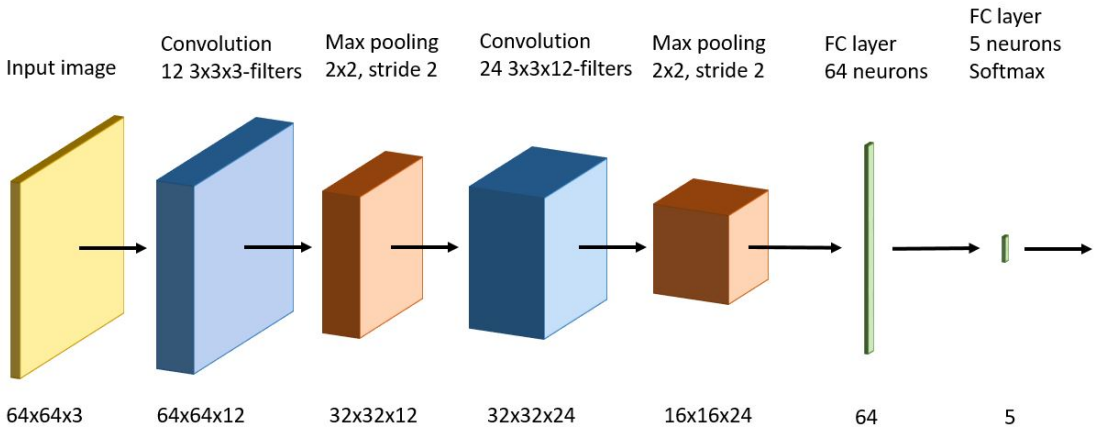


Figure 2.15: Example architecture of a convolutional neural network with different types of layers.

2.3.3.1 State of the Art – Residual Network

One type of network that has proven to be very successful for image classification is the residual neural network developed in 2015 [9]. A problem when stacking a lot of layers in an ordinary CNN is that the training error may increase when adding more layers to an already deep enough model. This should not happen as the end layers should just work as identity mappings and not affect the training error, as long as the model is not overfitting. The concept behind the residual network is to use blocks with convolutional layers where the input to each block is fed forward and added to the output. In this way the network is forced to learn a residual mapping instead of the identity mapping. This allows for training deeper networks with improved accuracy [9]. An example block of a residual network is shown in Figure 2.16.

In 2015 the authors of [9] won the ImageNet classification competition, ILSVRC, and variants of the residual network is still being used with great success. In this study the residual network is thus considered as state of the art when it comes to image classification.

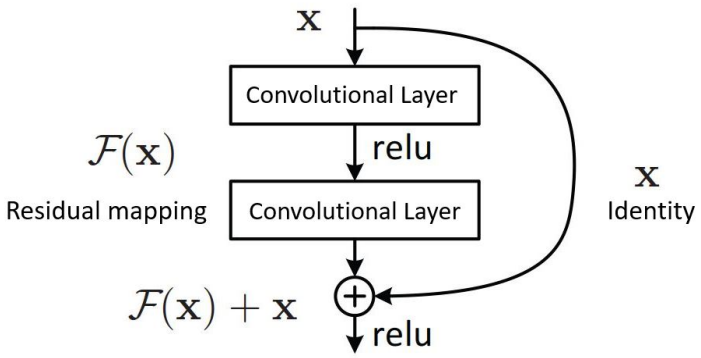


Figure 2.16: Example of a block in a residual network. The input x is fed both to the block and to the output of the block. Inside the block there are two convolutional layers that produces the residual mapping.

3
Methods

This chapter contains the methodology of starting with processed raw data from aircraft radar and establishing training data with two labels, clutter and target. Special attention is put on describing how synthetic targets are being constructed in Section 3.2.2, with regard to the size and behaviour of small boats. In Section 3.3 the methodology of finding the best suitable network architecture together with hyperparameter tuning is presented. Finally, different types of data parameters that are to be evaluated are presented in Section 3.4.

3.1 Processing of Raw Data and Peak Detection

The raw data used when creating training data is collected via an airborne radar, as described in Section 2.1.1, resulting in a range-Doppler plot of the search frame for each lobe position. In order for separation of objects in the two dimensions, the data is exposed to pulse compression and FFT. This data is hereafter called processed data and an example is shown in Figure 3.1 (A).

In the processed data, amplitude peaks are being detected in order to find sea clutter with characteristics similar to small boats. Detection of amplitude peaks is being performed with moving average by scanning the processed data in the range domain. For every pixel in the plot, a chosen number of pixels before and after are evaluated by calculating the mean of each region. If the pixel in between the two regions has a higher value than the mean values of the two regions, and is a local maximum, it is considered as a peak. These peaks also have to pass a set threshold to be considered a possible target. An example of detected amplitude peaks can be seen in Figure 3.1 (B).

3.2 Training Data

One of the most important things when training a neural network is to have reliable training data that is representative for the task that the network is going to perform. If the training set has shortcomings, e.g. is too small, does not contain enough variations and so on, the final result when evaluating the network might not be as good as expected.

In this project, the training data are labeled into two categories, sea clutter and target, making it a supervised learning type of problem. The training data are

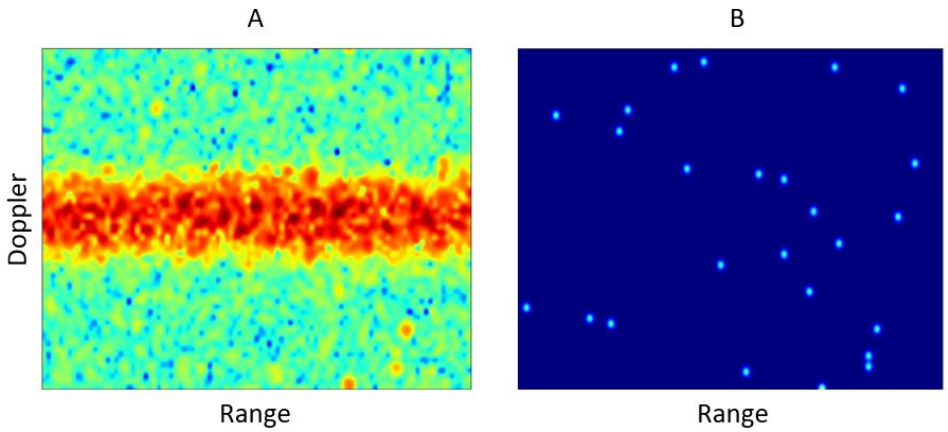


Figure 3.1: A: Example of processed radar data for one lobe position. B: Example of detected amplitude peaks in the data from (A).

collected by extracting patches from the processed data after peak detection, generating synthetic targets which are added to some of the patches and giving each patch a label. This is further described in Section 3.2.1 and 3.2.2. It is also important that the data set used for training is balanced, i.e. that the amount of data is equally distributed between the classes.

3.2.1 Sea Clutter

Sea clutter in the radar data can take many different shapes and amplitudes. Since the objective is to distinguish smaller boats from sea clutter, the training set need to contain clutter that is easily mistaken for such boats. This kind of clutter is found by setting the amplitude threshold high enough to only detect peaks that are above the noise level. Too large amplitudes are also dismissed since such peaks are not confused with the smaller boats of interest in this study.

Except for the amplitude criterion, there are other requirements as well. Peaks that are located beyond the horizon or too close to land are not considered. The reason for dismissing peaks close to land is that there is a risk of them being man-made objects such as buoys.

For each lobe position, the positions of the peaks that are detected and pass the criterion mentioned earlier are stored and a patch is extracted around each of these peaks. An example of such a detection is the black box shown in Figure 3.2, and the extracted patch can be seen in Figure 3.3 (A). This is done for several lobe positions in order to obtain a large amount of training data. It is unfortunately always a small risk that some of the patches labeled as clutter are actual boats, since the processed data is not labeled by Saab. All detected peaks are labeled as clutter and the proportion of actual boats among these labeled clutter samples is considered as being very low. It is thus believed that the training data is not biased too much to affect the result.

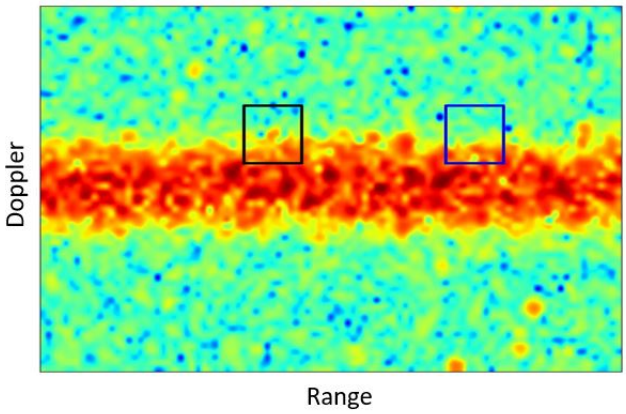


Figure 3.2: Example of two patches to be extracted. The black one with an amplitude peak in the center and the blue one without an amplitude peak in the center.

3.2.2 Synthetic Targets

For every peak that passes all criteria for the clutter case described in the previous section, one more patch with equal size is extracted around a position a given distance away in the same Doppler channel, see the blue box in Figure 3.2. This patch represents sea clutter without any center peak and acts as a background that a synthetic target later is added to. An example of an extracted background patch is seen in Figure 3.3 (B). Since this is done each time a clutter patch is extracted, the training data set contains the same amount of patches representing clutter and targets.

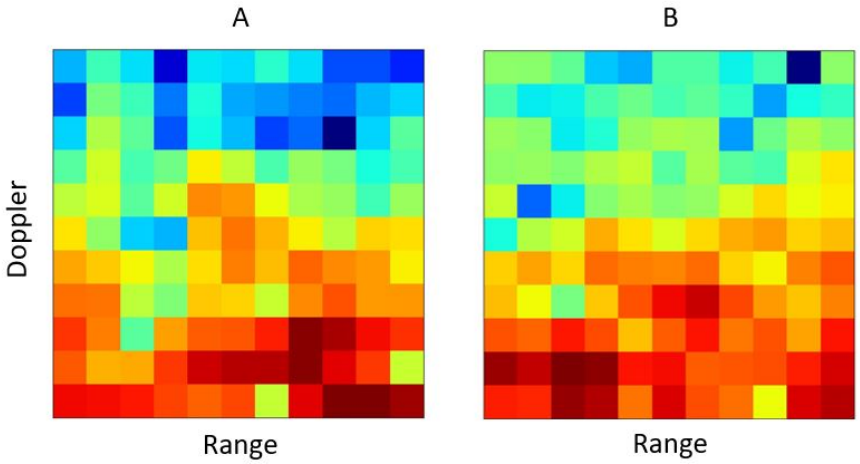


Figure 3.3: A: Extracted clutter patch with an amplitude peak in the center. B: Extracted patch without an amplitude peak in the center. Note that the figures do not share color scale, meaning that the maximum value in each patch will always be dark red and the minimum value dark blue, regardless of the actual values.

The data representing targets is generated synthetically, i.e. a mathematical model is used to generate radar echoes resembling echoes from real targets. To obtain re-

sults that can be trusted, it is of great importance that these targets look realistic. Dammert [10] has written a paper that describes how these kind of targets can be modeled, which is the basis for the target implementation in this study.

A vessel traveling through sea has 3D dynamics, i.e. roll, pitch and yaw motions. In this study it is assumed that the pitch and roll motions have less impact on the received echo than the yaw motion, since they give rise to lower motion in the direction of the radar. The yaw motions will make certain parts on the vessel move away from or towards the radar and may therefor result in some spread over Doppler channels. For that reason only the yaw motion is taken into account when generating the targets, making the dynamic modelling a 2D problem.

The received echo from a target is the sum of echoes from several different scatterers, where a scatterer represents an object on the target, for example a corner or a flat plate [3]. The vector representing the total return can be described as

$$z = \sum_{n=1}^{N} A_n \cdot e^{j\phi_n} \cdot e^{j(v_{shift}+\Delta v_{shift,n})p} \tag{3.1}$$

where A_n is the amplitude, ϕ_n is the phase shift due to the difference in distance between the scatterer and the centre of the target. v_{shift} is the phase of the target centre and $\Delta v_{shift,n}$ is the phase shift due to the difference in velocity compared to the centre. p is a vector containing all transmitted pulses.

The targets of interest in this study are boats in the size range from a jet ski to a professional fishing boat. The length, L, is uniformly randomized to suit the range specified. The width of the targets is set to be $L/3$. Each target is generated as a rectangle, representing a boat seen from above, within these dimensions. The different scatterers are randomly placed within the rectangle resembling the target. Since a boat can travel in all directions, all points are also rotated around the targets centre according to the rotation matrix seen in Equation 3.2.

$$R = \begin{bmatrix} cos\ \alpha & -sin\ \alpha \\ sin\ \alpha & cos\ \alpha \end{bmatrix} \tag{3.2}$$

where α is randomly chosen between 0°and ±180°. An example of a target can be seen in Figure 3.4.

Both the distances and the velocities for the different scatterers are calculated in two dimensions, i.e. the height difference is not taken into account. Also, only the range and velocity component in the direction of the radar is considered. For each scatterer, the distance and velocity is calculated, in accordance with Figure 3.5, as

$$d = r cos\ \beta$$
$$v_x = r\omega sin\ \beta$$

The scatterer density is set in accordance with [10], where a target has both larger and smaller scatterers. The amount of large scatterers is uniformly randomized to

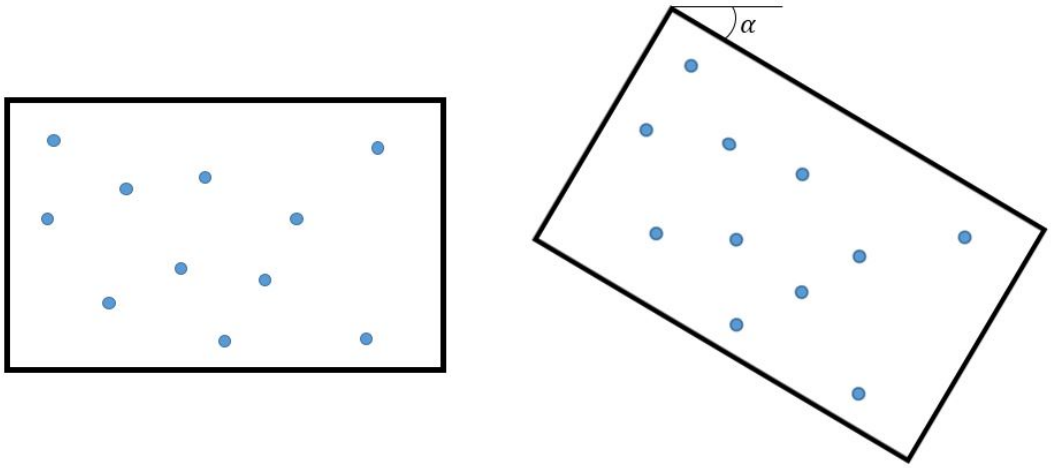


Figure 3.4: A generated target with randomly placed scatterers. To the left in original position, to the right rotated with an angle α.

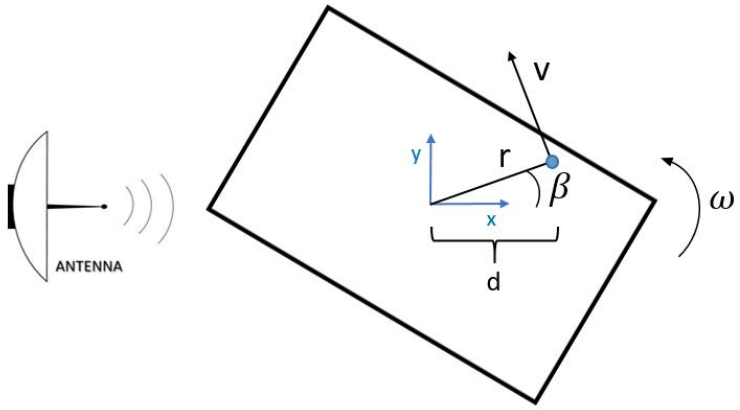


Figure 3.5: Visualization of the distance from the scatterer to the centre point of the boat together with the trigonometry behind the velocity component as an effect of the angular velocity ω.

be one per 8-10 square meters, and the amount of small scatterers is uniformly randomized to be 2-3 per square meter.

The return from a boat that has an angular velocity may spread over several Doppler channels. Here, that spread is set to be over a maximum of three channels since more than that is not considered normal for these kind of targets. The acceleration is calculated according to

$$a_{max} = \left(\frac{\lambda N}{\Delta T}\right)^2 \cdot \frac{1}{8L}$$

where λ is the wave length of the pulses, N the spread in Doppler channels and ΔT the integration dwell time. N is assigned values 1, 2 or 3 with frequency distribution 80%, 15% and 5% respectively, which is considered as a good approximation of a small boat's characteristics. The acceleration, a, for each target is randomly chosen

between 0 and a_{max}. The angular velocity is

$$\omega = \sqrt{\frac{a}{L/2}}$$

The total return described in Equation (3.1) can be divided into two summations as

$$z = \sum_{n=1}^{N} A_n \cdot e^{j\phi_n} \cdot e^{j(v_{shift}+\Delta v_{shift,n})p} + \sum_{m=1}^{M} A_m \cdot e^{j\phi_m} \cdot e^{j(v_{shift}+\Delta v_{shift,m})p}$$

where N and M are governed by the two different scatterer densities.

The targets are transformed from time domain to Doppler domain in the same way as the processed data, where an FFT is performed. The length of the target vector corresponds to the number of pulses used by the radar thus generating an equally long vector in the Doppler domain. A smaller vector, x, is then extracted around the centre of the target, having the same length as a patch side.

In order to obtain a good representation of the range domain of the target, a model is used based on pulse compression of a signal. Barton [11] shows how a radar waveform is compressed with the help of a filter. The filter is structured with a main lobe and smaller side lobes. However, the amplitudes of the side lobes in this project will be suppressed by the background noise since only small boats are considered. The approximation in Equation 3.3 is a simplified version of the same methodology as in [11].

$$y = 10^{-1.2((r+r_{shift})/R3)^2} \tag{3.3}$$

where r is the interval of range bins for the patch, with the centre bin being equal to zero, r_{shift} is the difference in range compared to the centre and $R3$ is a constant defined by the working mode of the radar. A visualization of y as a function of r can be seen in Figure 3.6.

The synthetic target is generated by multiplying the vectors x and y to form a matrix with the same size as an extracted patch. The obtained synthetic targets are expressed in the complex domain and added to the background patch before converted to decibels. In Figure 3.7, the background patch (A), a generated synthetic target (B) and the final patch (C) are shown. Note that all patches are visualized in decibels, and the final target patch (C) may not look like expected since the imaginary parts of (A) and (B) can strengthen or reduce the amplitude.

The final target also has to pass the same threshold as the detected clutter, making sure that it would have been detected as a possible target if it was placed in the original data.

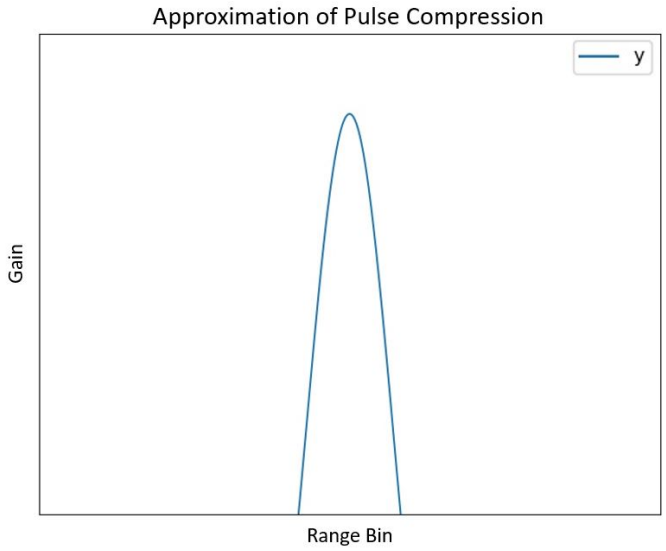


Figure 3.6: Approximation of pulse compression in the range domain of the target.

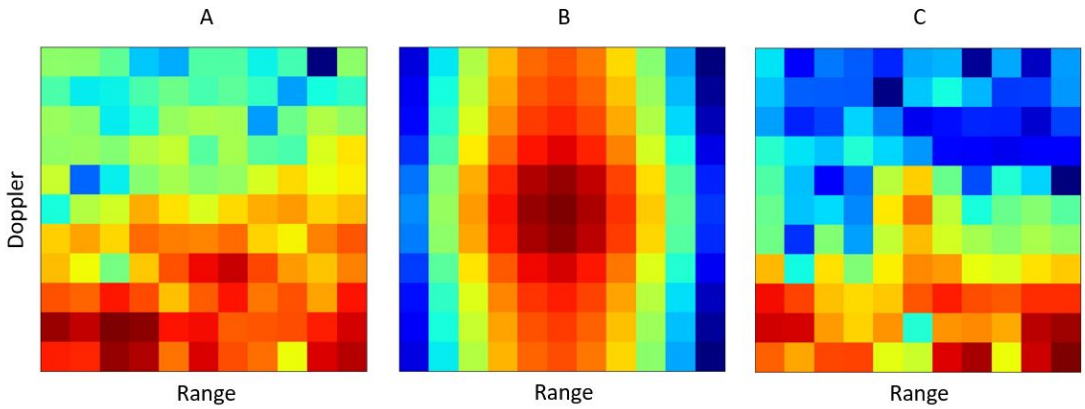


Figure 3.7: A: Patch where a synthetic target is to be added. B: A generated synthetic target. C: Synthetic target and background added together, resulting in a patch labeled as target.

3.3 Network

A logarithmic operation is performed on the absolute value of the complex radar data. The data used in this study is thus structured similar to images with one color channel, but with one difference. Instead of pixel values describing how bright each pixel is, each value in the data represents an amplitude in decibels. Since this similarity exists, the problem is approached as an image classification task where the convolutional neural network has proven to best performing [8]. This project investigates how a convolutional neural network can be implemented to suit this particular problem, where different architectures along with different hyperparameters are evaluated.

The measurement of how good a specific architecture or hyperparameter choice

perform is evaluated by studying the classification accuracy of the validation set, according to Equation 3.4. This measurement is used until a final best performing network has been chosen, together with a proper training data size and patch size. Another interesting measurement to evaluate, when studying a binary classifier, is to observe the Receiver Operating Characteristic curve (ROC curve). The ROC curve used in this project show the true positive rate as a function of false alarms per second. The true positive rate is equal to the percentage of how many of the targets that are actually classified as targets and false alarms are false positives that in fact are clutter but are predicted as targets. The ROC measurement is used as an additional evaluation of the performance, since it is conventionally used in radar context.

$$\text{Classification Accuracy} = \frac{\text{Amount of Correctly Classified Samples}}{\text{Total Amount of Samples}} \tag{3.4}$$

3.3.1 Preprocessing of Input Data

The input data to the neural network is preprocessed. A common practice is to subtract the mean value of the data set from each individual value in order to zero-center the data and then normalize in some way [8]. A couple of different normalization techniques were evaluated and the best result was achieved using the z-score normalization as shown in Equation 3.5, where the data are divided by the standard deviation after it has been zero-centered.

$$X_{norm} = \frac{X - \text{mean}(S)}{\text{std}(S)} \tag{3.5}$$

The mean and standard deviation are computed from the training data set and the same values are used when preprocessing the test data.

3.3.2 Hyperparameters and Architecture

The evaluation of different architectures and hyperparameters in a network is inspired by the grid search methodology from [12], which is one way to evaluate hyperparameters structurally. There exist numerous hyperparameters outside of this list, but these are considered as a good start to evaluate. The idea is to select one or two hyperparameters and train a shallow network consisting of only one convolutional layer and an FC layer outputting the probability of being a target. Different values for the selected hyperparameters are evaluated and the ones giving the best result are picked for further tests. This is then repeated for the next parameters to be evaluated. When all different hyperparameters have gone through the testing, one combination will stand out as the best and act as a base for further architectural evaluation.

3.3.2.1 Hyperparameters

The hyperparameters and the values that will be evaluated are seen in Table 3.1.

The optimizers shown in Table 3.1 are all using GD in some extent. All of the optimizers are evaluated together with the default parameters provided by [13]. The optimizer SGD is evaluated once more, together with different settings of learning rate and momentum since it is not adaptive as the other optimizers are [14].

Different initialization of the weights in the network's convolutional layer are evaluated. The weight initializers chosen to evaluate are the same as the ones listed in [12], which are available from [13].

The activation functions shown in Table 3.1 are applied to the convolutional layer one at a time. The FC layer is set to always have a softmax, since its purpose is to output probabilities of the two categories clutter and target. Keras [13] provide the listed activation functions.

The number of epochs used when training is not evaluated as a hyperparameter. Instead a function called early stopping is used, which stops the training when the validation loss has stagnated. In this way the training time is adapted to be long enough for each model, and it also prevents overfitting which may be a result of training a model for too long.

Table 3.1: Hyperparameter types together with values or functions to evaluate.

Kernel Size	1x1	2x2	3x3	
Stride	1	2	3	
Batch Size	16 256	32	64	128
Optimizer	SGD Adamax	RMSprop Nadam	Adagrad	Adam
Learning Rate (SGD)	0.0001 0.2	0.001 0.3	0.01	0.1
Momentum (SGD)	0.0 0.8	0.2 0.9	0.4	0.6
Weight Initializer	uniform glorot normal	lecun uniform glorot uniform	normal he normal	zeros he uniform
Activation Function	softmax tanh	softplus sigmoid	softsign hard sigmoid	relu linear

After the best hyperparameter settings are found, the architecture of the network is evaluated.

3.3.2.2 Architecture

The first step of figuring out different network architectures is made by randomizing the number of layers and filters, as seen in Table 3.2. The number of convolutional layers are restricted to not be more than five layers deep. This restriction is set in order for the training not to be too time consuming. By randomizing, the huge variation of evaluating all possible combinations is avoided, thus saving time of calculations that would have taken many days. In total 75 models were evaluated, which is a number picked by letting the number of layers act as a base of how many models to evaluate. This is due to that the more layers the model contain the more combinations of filter sizes to evaluate. If the number of layers is one, five models are randomized, if the number of layers is two, ten models are randomized and so on.

Table 3.2: Architecture features together with parameters to evaluate.

Number of Convolutional Layers	1	2	3	4	5		
Number of Filters	2	4	8	16	32	64	128

In the next step of finding the best model, max pooling is to be evaluated. When max pooling is added, the kernel size used is 2x2 and the stride is set to be one. Since the models are relatively deep, the regularization techniques dropout and weight constraint are evaluated in order to avoid potential overfitting. Another randomization process is performed. Max pooling and dropout are randomly added after each convolutional layer, starting with the second layer, and the weight constraints are randomized for each convolutional layer. The dropout ratio is randomized between 0.1 and 0.5 and the weight constraints are assigned integer values from 1 to 5. In total 250 models were randomized and evaluated. Learning rate decay was used throughout testing with a reduction of ten percent every fifth epoch, starting with a learning rate of 0.001. After this test, the best performing model is chosen.

3.3.3 Residual Network

A comparison between the best performing model from the previous section with variations of the state of the art network, regarding image classification, is performed. The comparison is made with varying architecture and hyperparameter setting of the residual network, such as depth and number of filters. This is performed in order to evaluate if the residual network may outperform the regular CNN developed in previous section.

3.4 Data parameters

Since many flight tests have been recorded and each flight generates a lot of data, the amount of processed data available is enormous. Different flights generate data from different types of weather conditions. Each flight may use one of three modes, where the mode is equal to the number of pulses used when collecting the data. In

order to find what impact the mode and shape of the training data have, some data parameters are evaluated.

The three modes generate different Doppler resolution, which may be crucial to the classifier, and is thereby evaluated. The modes are evaluated together with varying sizes of extracted patches. A fixed patch size of 11x11 is used when extracting the patches from the processed data. This size is considered as relatively large since a target or clutter peak is smaller than this and should not affect the utmost frames of pixels. Evaluating the performance of the classifier based on patch size is then performed by cropping the utmost frame of pixels of the patch. The amount of samples in the training data, also known as training data size, varies depending on how many targets that are generated for each of the data sets, where a data set consists of data from one flight. The current available amount of extracted training data is used for each of the data sets when evaluating different patch sizes. Data from only one flight are used when training and validating the accuracy in order to simplify testing. It is possible to use data from different flights, but the simplification using one flight is believed to not harm the process of evaluating the effect of the patch size.

Another thing evaluate the performance of the classifier with is to try different sizes of the training data set. By starting of with extracting a large amount of training data the amount is halved ten times. The validation accuracy is evaluated for each of the listed training data sizes leading to a proper amount to use.

In order to find what pixels of the patches of clutter and targets that give the most information to the classifier, masking is used. Masking is performed to confirm that the synthetic targets are designed realistically and are not affecting the background in an undesirable way. The masked out pixels are assigned the same fixed value of 10.0, which is outside of the normalized pixel values interval and should thus not affect the unmasked pixels. Since masking generates the same information for all of the masked out pixels there is no way of predicting a correct label based on these pixels. The masking procedure is performed in four different ways. The first way is to mask from the centre of the patch and outwards, later on called center masking. The second way is to mask in the opposite direction, namely from the utmost frames of pixels of the patch inwards, which is called inverted masking. The third and fourth ways are to mask bands of pixels in the range and Doppler domain respectively, called range masking and Doppler masking.

The parameters used when evaluating the training data can be seen in Table 3.3. The modes differ in the number of pulses used by the radar, where Mode 1 uses the least and Mode 3 the most number of pulses. The values in the row "Training Data Size" is the total amount of training samples. Since there are equally many clutter and target samples, half of the amount are clutter and half are targets. The last row, "Masking Type", is written with letters and numbers. The letters determine whether the masking is center wise, inverted, range wise or Doppler wise. The numbers determine how much from the patch that is masked. If it is center masking, a one means that the centre pixel is masked and a three that 3x3 in the

centre of the patch is masked. For the inverted masking approach, a two means that the two utmost frames of pixels are masked. For the range and Doppler kind of masking, a one is simply a row or column masked and a three is three rows or columns masked around the center pixel. Examples of all of the masking operations presented in Table 3.3 are shown in Figure 3.8.

Table 3.3: Data Parameters together with values and structures to evaluate.

Mode	1	2	3				
Patch Size	11x11	9x9	7x7	5x5	3x3		
Training Data Size	500,000 7,812	250,000 3,906	125,000 1,953	62,500 976	31,250 488	15,625	
Masking	C,1	C,3	I,2	R,1	R,3	D,1	D,3

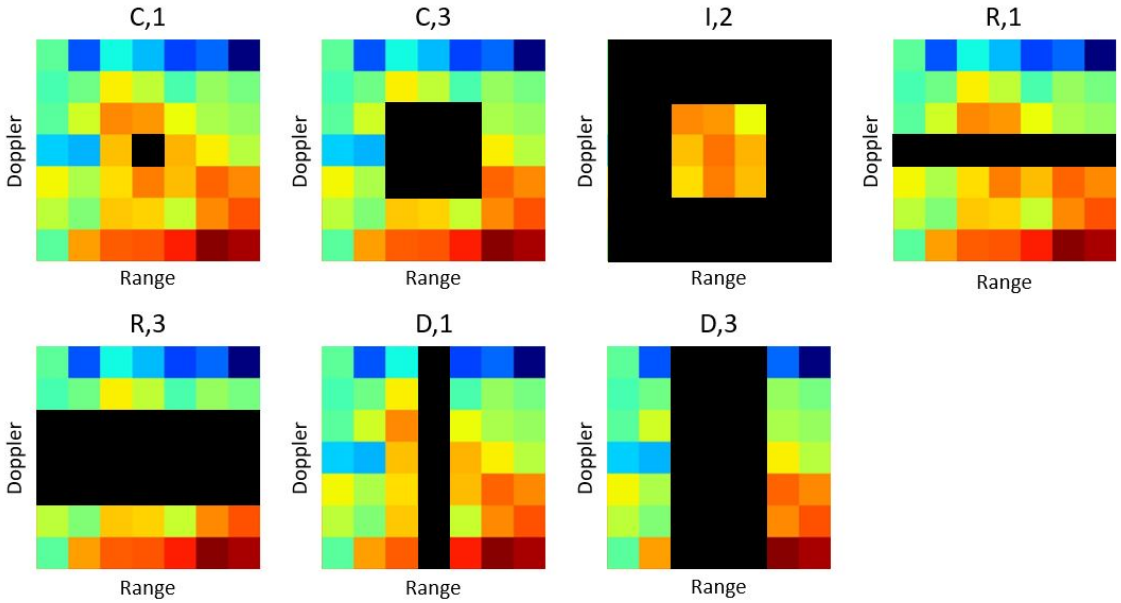


Figure 3.8: Examples of masking operations on a 7x7 patch of sea clutter.

3.4.1 Extended Evaluation

In previous evaluation steps, the performance of certain models are evaluated by using data from the same flight as when training. It may be of interest to extend the evaluation of a model by testing it on data from other flights, both from the same mode as well as another mode. The extended evaluation also includes a ROC curve where another type of measurement is tested.

3.4.1.1 Crossevaluation of Models

In the crossevaluation of models, data from two flights from each of the three modes are used to train models. As stated earlier, there are three available modes, corresponding to the number of pulses used. Each model is trained and tested on data

from the same flight as the training is based on. The model is also tested on data from other flights, both from the same mode and from the other two modes, to see how well it generalizes.

3.4.1.2 Final Training

In the final training, the final model is trained with as much data as available. For Mode 1 data from one flight is used for training, for Mode 2 three flights are used and for Mode 3 four flights are used. There are different amount of flights used for training because it takes a lot longer time to extract similarly many samples the lower the model mode is.

3.4.1.3 ROC Curve

In radar context the ROC curve is conventionally used, in order to obtain a feeling of whether the number of false alarms per second is manageable or not by the operator. The ROC curve is plotted with values of the confusion matrix based on results from the final training. A confusion matrix is a two dimensional table that shows the true and predicted labels of classified data, as shown in Figure 3.9. TN is the abbreviation for true negatives, FP for false positives, FN for false negatives and TP for true positives. The sum of each row is the total amount of target and clutter samples used in the test set.

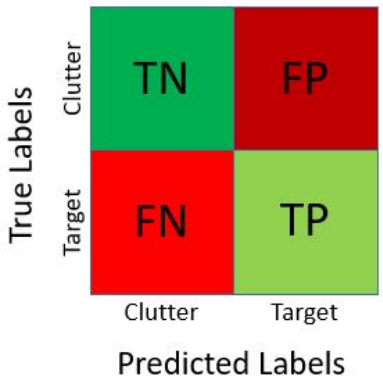


Figure 3.9: Structure of the confusion matrix used in this study.

The ROC curve used in this study is obtained by measuring how TP varies when shifting the classification threshold. The default probability threshold when using a softmax as activation function is 0.5. In this study, if the output is lower than 0.5 it is predicted as a clutter otherwise it is predicted as a target. The higher the threshold of the softmax function is set, the higher the confidence that the samples labeled as target actually are targets thus resulting in fewer false alarms per second. The downside of choosing a high threshold is that TP gets lower as the threshold increases leading to a lower total classification accuracy. The plotted ROC curve is obtained by letting the threshold increase from 0.0 to 1.0. Equation 3.6 shows how

the number of false alarms per second is obtained for each threshold value, where the total time is calculated as presented in 3.7.

$$\text{false alarms per second} = \frac{\text{FP}}{\text{total time}} \tag{3.6}$$

$$\text{total time} = \frac{\text{lobe positions} \cdot \text{number of pulses}}{\text{pulses per second}} \tag{3.7}$$

4
Results and Discussion

This chapter shows the results based on the implemented methodologies described in the previous chapter. A discussion of the findings is also presented for each of the sections. The work is performed in Python language using the libraries of Tensorflow [15] and the deep learning library Keras [13].

4.1 Training Data

Example of how the generated training data can look like is shown in Figure 4.1 and in Figure 4.2 it is visualized how the amplitudes of the center pixel for the extracted patches are distributed. In the first histogram a sharp edge is seen as upper limit. This is due to the threshold of neglecting large boats. If the amplitude of the extracted clutters is larger than this, it is easier for the classifier to predict the label since the amplitude of the synthetic targets are bounded to be smaller than this fixed amplitude. The third histogram shows how the lower amplitudes from the synthetic targets, shown in the second histogram, are being neglected. The reason for this is because the synthetic targets are being exposed to the same threshold as the clutter, i.e. having to be large enough to be detected from the surrounding noise. It can also be seen that the distributions of the amplitudes of the synthetic targets after threshold is similar to that of the clutter's. This is good since it reflects the reality where the small boats are similarly looking as the clutter.

4.2 Network

In this section, results related to hyperparameter tuning and architectural structure of a network are presented.

4.2.1 Hyperparameters and Architecture

When performing small scaled tests, it is found that a kernel size of 3x3 and a stride of 1 give the best result. The following tests are thereby evaluated with these settings. As the patches are relatively small, zero-padding is used in order to preserve the spatial size of the patch after each convolutional layer. The results presented in this section are obtained using data from one flight with Mode 3 and the amount of training samples is approximately 320,000.

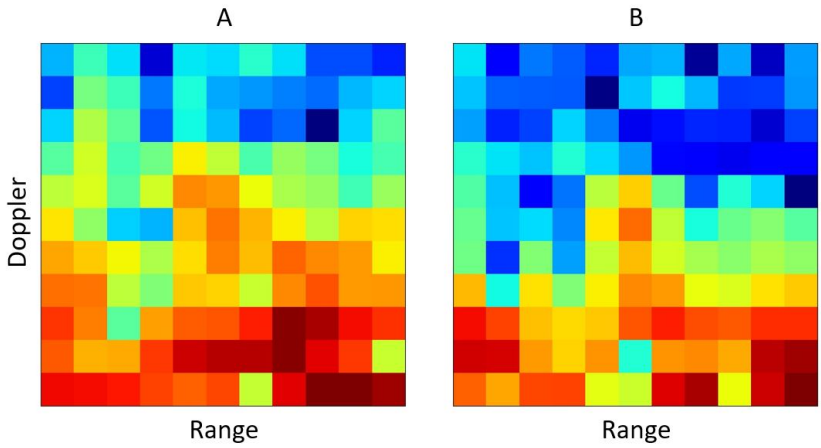


Figure 4.1: Example of generated training data with a patch size of 11x11. A: Patch with clutter. B: Patch with synthetic target. Note that the figures do not share color scale, meaning that the maximum value in each patch will always be dark red and the minimum value dark blue, regardless of the actual values.

4.2.1.1 Hyperparameters

As can be seen in Table 4.1, a smaller batch size gives slightly better results but the performance differ very little. Since the batch size does not have a great impact on the performance, and a larger batch size speeds up the computations with the computational capacity available when performing this project, a batch size of 128 is chosen to continue with.

Table 4.1: Classification accuracy for different batch sizes.

Batch Size	16	32	64	128	256
Accuracy	0.908	0.908	0.907	0.905	0.903

Table 4.2 shows the performance with different optimizers. Here Adam gives the best result, although with a small margin. The default SGD optimizer shows quite poor results, but when evaluated with different learning rates and momentum, as seen in Table 4.3, its performance increases. By looking at the complete result list in Table A.1 in Appendix A it can be seen that the learning rate has a bigger impact than the momentum in this study. Even though the performance of the SGD optimizer increases with a different learning rate, Adam is still the best and is therefor chosen to be used in further testing.

Table 4.2: Classification accuracy with different optimizers.

Optimizer	SGD	RMSprop	Adagard	Adam	Adamax	Nadam
Accuracy	0.886	0.906	0.861	0.907	0.903	0.906

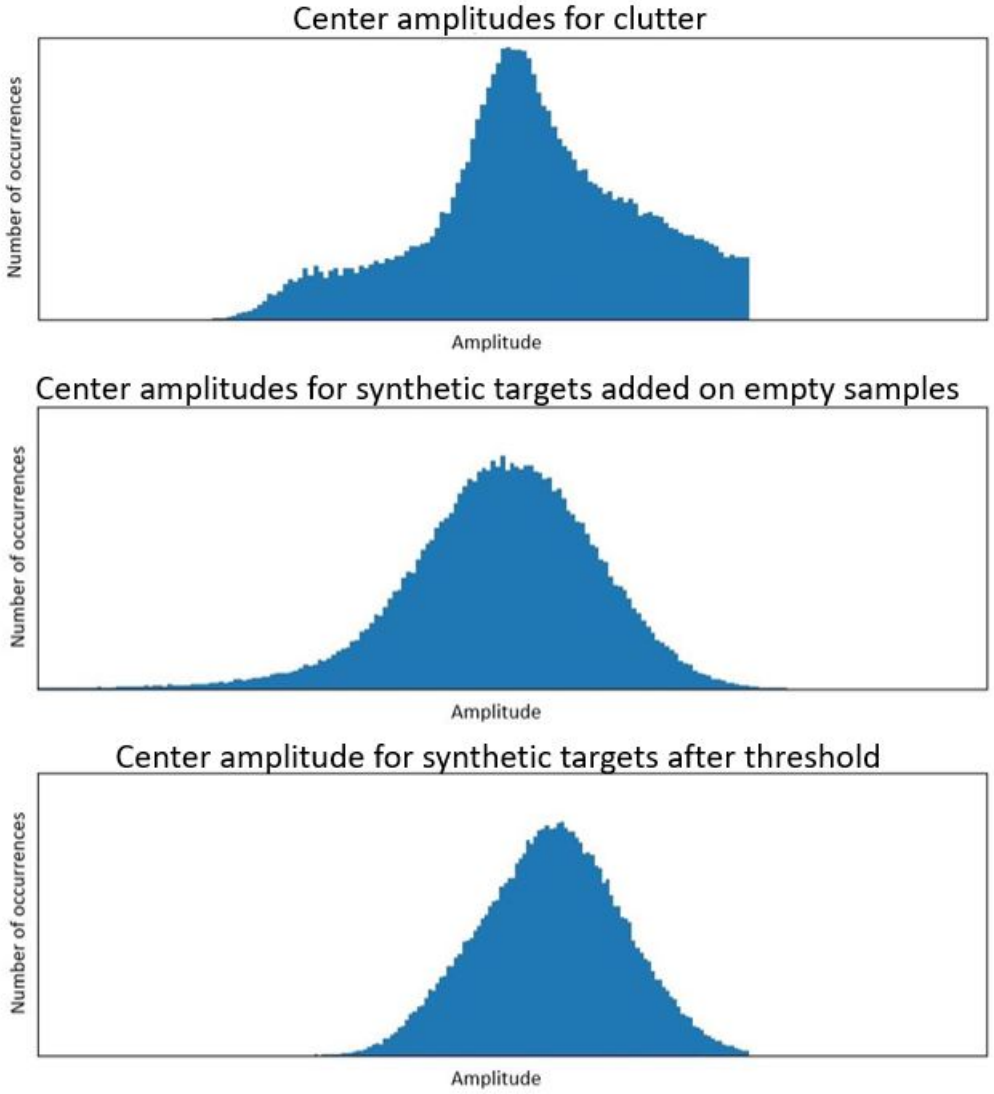


Figure 4.2: Distribution of amplitudes of center pixel from extracted patches for clutter and synthetic targets. The amplitudes of the synthetic targets are presented before and after thresholds.

Table 4.3: The top five classification accuracies for SGD as optimizer with different learning rates and momentum. The complete table of tested learning rates and momentum are shown in Section A.1 in Appendix A.

Learning Rate	0.001	0.01	0.01	0.01	0.01
Momentum	0.9	0.0	0.2	0.4	0.6
Accuracy	0.905	0.905	0.904	0.905	0.905

The results in Tables 4.4 and 4.5 shows that the weight initializer "uniform" performs the best and the best performing activation function is "relu". Both of the parameters win with small margins, but are still chosen to be used for further experiments.

Table 4.4: Classification accuracy for different types of weight initialization.

Weight initialization	uniform	lecun uniform	normal	zeros
Accuracy	0.907	0.905	0.901	0.498
Weight initialization	glorot normal	glorot uniform	he normal	he uniform
Accuracy	0.906	0.906	0.905	0.906

Table 4.5: Classification accuracy for different activation functions.

Activation Function	softmax	softplus	softsign	relu
Accuracy	0.905	0.898	0.903	0.906
Activation Function	tanh	sigmoid	hard sigmoid	linear
Accuracy	0.901	0.903	0.897	0.869

4.2.1.2 Architecture

The best performing models are shown to be built up by four or more convolutional layers. The top three models out of the 250 generated, based on classification accuracy, are shown in Table 4.6. As can be seen, the performance of the top three models are equal. However, the number of parameters varies much. Since it is computationally more efficient with fewer parameters to optimize, Model 3 is chosen as the best performing model. The structure of Model 3 is visualized in Figure 4.3.

Table 4.6: The top three models with best classification accuracies, with regard to different number of layers, filters, max pooling and dropout. The amount of parameters for each model is also presented.

Model	Model 1	Model 2	Model 3
Layer type	Conv 64 Conv 128 Dropout 0.4 Conv 32 Conv 32 Max pooling Dropout 0.3 Conv 32 Max pooling Flatten FC 16 FC 2	Conv 64 Conv 64 Dropout 0.2 Conv 32 Max pooling Conv 64 Dropout 0.5 Conv 32 Dropout 0.4 Flatten FC 2	Conv 16 Conv 16 Conv 64 Conv 32 Flatten Dropout 0.2 FC 2
Accuracy	0.921	0.921	0.921
Parameters	142,738	95,298	33,362

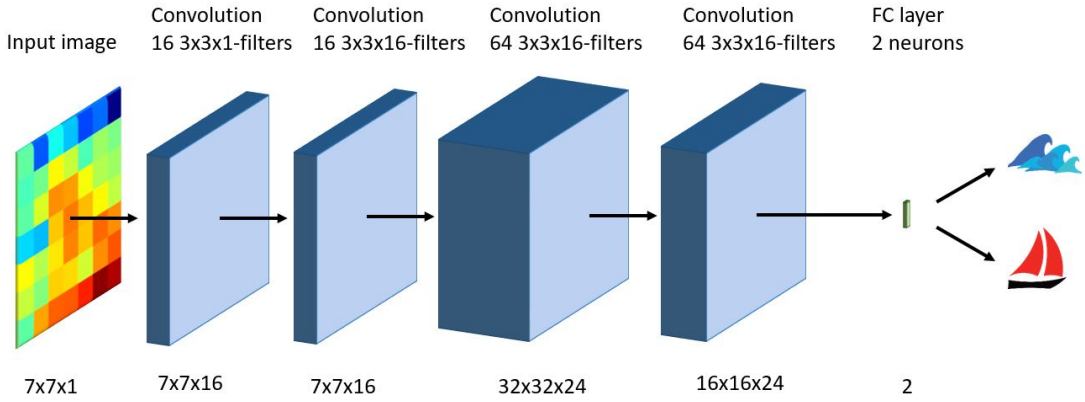


Figure 4.3: Best performing model with example of input image in the form of a 7x7 patch and the two classification categories clutter [1] and boat [2].

4.2.2 Residual Network

The best performing residual network is shown in Table 4.7, where a block is structured in the same way as in Figure 2.16. The number after each block represents the number of filters in the convolutional layers. After each convolutional layer there is a batch normalization present and before flattening and the final FC layer there is an average pooling layer. Both batch normalization and average pooling are inspired by the model from [9].

It can be seen that the residual network achieve a classification accuracy of 91.9%, which is not above the classification accuracy of the models shown in Table 4.6. The number of parameters is also a lot higher than the models in Table 4.6. Because of these reasons, the residual network is not used for further experiments.

Table 4.7: Classification accuracy and structure of best performing residual network.

Model	Residual Network
Layer type	Conv 32 Block 32 Block 32 Block 64 Block 64 Block 64 Block 64 Average pooling Flatten FC 2
Accuracy	0.919
Parameters	320,226

4.3 Data Parameters

Table 4.8 shows how the classification accuracy is similar for 11x11, 9x9 and 7x7 sized patches. After 7x7 the classification accuracy tends to decrease a bit. Since it is faster to train and evaluate a model with smaller training data, 7x7 is the patch size chosen for further experiments.

Table 4.8: Classification accuracy for different patch sizes and modes.

Mode	1				
Training Data Size	315,694				
Patch Size	11x11	9x9	7x7	5x5	3x3
Accuracy	0.919	0.921	0.919	0.911	0.885
Mode	2				
Training Data Size	293,112				
Patch Size	11x11	9x9	7x7	5x5	3x3
Accuracy	0.933	0.935	0.934	0.927	0.906
Mode	3				
Training Data Size	321,342				
Patch Size	11x11	9x9	7x7	5x5	3x3
Accuracy	0.956	0.956	0.955	0.949	0.924

It can be seen in Table 4.9 that the classification accuracy decreases as the training data size decrease. In order to speed up computations in future experiments, the size 125,000 is chosen. This size is relatively small, but does not loose to much in performance in comparison to the larger training data sizes and is thus considered as a good compromise between computational efficiency and performance.

Table 4.9: Classification accuracy for different amount of training data and modes.

Mode	1	2	3
Training Data Size	Accuracy	Accuracy	Accuracy
500,000	0.955	0.935	0.921
250,000	0.952	0.935	0.920
125,000	0.952	0.933	0.916
62,500	0.951	0.930	0.913
31,250	0.947	0.929	0.907
15,625	0.942	0.917	0.893
7,812	0.923	0.914	0.891
3,906	0.898	0.905	0.862
1,953	0.843	0.836	0.831
976	0.756	0.872	0.815
488	0.753	0.670	0.629

When experimenting with the masking operations, a training data size of 125,000 is used throughout the test based on the results from Table 4.9. The classification

accuracies of the masking operations for each of the data sets are shown in Table 4.10. It can be seen that the 3x3 pixels in the patch give the most information for the classifier, since the classification accuracies when these 3x3 pixels are masked out generate the lowest results. Furthermore, masking one column (Doppler wise masking) gives approximately ten percentage points worse classification accuracy than masking one row (range wise masking). Masking three rows or columns seem to have the similar effect as masking the 3x3 centre pixels, although slightly less information. Again, Doppler wise masking is a little worse than range wise masking.

Table 4.10: Classification accuracy for different masking operations and modes.

Mode	1						
Masking	C,1	C,3	I,2	R,1	R,3	D,1	D,3
Accuracy	0.925	0.642	0.927	0.902	0.626	0.825	0.617
Mode	2						
Masking	C,1	C,3	I,2	R,1	R,3	D,1	D,3
Accuracy	0.897	0.596	0.901	0.871	0.574	0.772	0.587
Mode	3						
Masking	C,1	C,3	I,2	R,1	R,3	D,1	D,3
Accuracy	0.876	0.635	0.877	0.848	0.624	0.745	0.606

4.3.1 Extended Evaluation

This section includes training of models with data from several flights and testing on data from another flight. A plot of ROC curves for each of the finally trained models is also presented.

4.3.1.1 Crossevaluation of Models

In Table 4.11 the column "Model Name" states which mode that is used when training and the index 1 or 2 define whether the data is from the first or second flight from the specified mode. The same definition holds for the column "Test Data", where the mode and index state where the test data is extracted from. A training data size of 125,000 is used throughout this test.

By observing the results from Table 4.11 it can be seen that the results of predicting unseen data is the best when using a model trained on data from the same flight. This is reasonable since the clutter level caused by weather conditions are similar in training data and test data. However, the result of using a trained model from a certain mode still operates properly when predicting test data from another flight with the same mode. The results also show that it is easier for a model trained on data from Mode 2 to predict test data from Mode 3 and vice versa, than it is for a model trained on Mode 1 data to predict test data from Mode 2 and 3. The same holds in the opposite direction, where it is harder for models trained on Mode 2 or 3 to predict test data from Mode 1.

Table 4.11: Classification accuracy on test data based on model mode of predictor and mode of test data.

Model Name	Test Data	Accuracy
Model 1.1	1.1	0.952
Model 1.1	1.2	0.926
Model 1.1	2.1	0.848
Model 1.1	3.1	0.822
Model 1.2	1.2	0.941
Model 1.2	1.1	0.942
Model 2.1	1.1	0.823
Model 2.1	2.1	0.933
Model 2.1	2.2	0.934
Model 2.1	3.1	0.910

Model Name	Test Data	Accuracy
Model 2.2	2.2	0.944
Model 2.2	2.1	0.924
Model 3.1	1.1	0.838
Model 3.1	2.1	0.927
Model 3.1	3.1	0.916
Model 3.1	3.2	0.939
Model 3.2	3.2	0.942
Model 3.2	3.1	0.911

4.3.1.2 Final Training

The column "Model Name" in Table 4.12 define which mode that is used and what flights that the data used for training are collected from. The same notation holds for the column "Test Data". The training data size is increased to 500,000 when training the models. It can be seen that the accuracy is still high when using data from many flights, in comparison with using data from only one flight. Model 1.1 achieves a lower classification accuracy than the other two models. One reason of this may be due to that it is only trained on data from one flight and is therefor not as generalizing as the other two models.

Table 4.12: Classification accuracy on test data from one flight based on model trained on one or more other flights from the same mode.

Model Name	Test Data	Accuracy
Model 1.1	1.2	0.926
Model 2.1-3	2.4	0.948
Model 3.1-4	3.5	0.947

4.3.1.3 ROC Curve

Examples of confusion matrices for Mode 3 are shown in Figure 4.4. Similar matrices have been conducted for all modes and for probability thresholds ranging from 0.0 to 1.0. The ROC curves obtained for the final models, which are presented in Section 4.3.1.2, can be seen in Figure 4.5. The figure shows that all models follow the same pattern, with increasing true positive rate as the false alarm rate increases. It can be seen that the true positive rate reaches 1.0 and stagnates when the false alarm rate reaches approximately 400 for all models. It can also be stated that the classification accuracy decreases as the false alarm rate decreases, as seen in Figure

4.4.

The ROC curves correlate to the achieved classification accuracies in Table 4.12. A model with lower classification accuracy requires a higher false alarm rate to achieve the same true positive rate compared to a model with higher classification accuracy.

For a real time system, the number of false alarms per second can not be too high since it would be unmanageable for the operator. By adjusting the probability threshold for the classifier, the false alarm rate can be decreased according to the ROC curves to obtain a reasonable level decided by the operator.

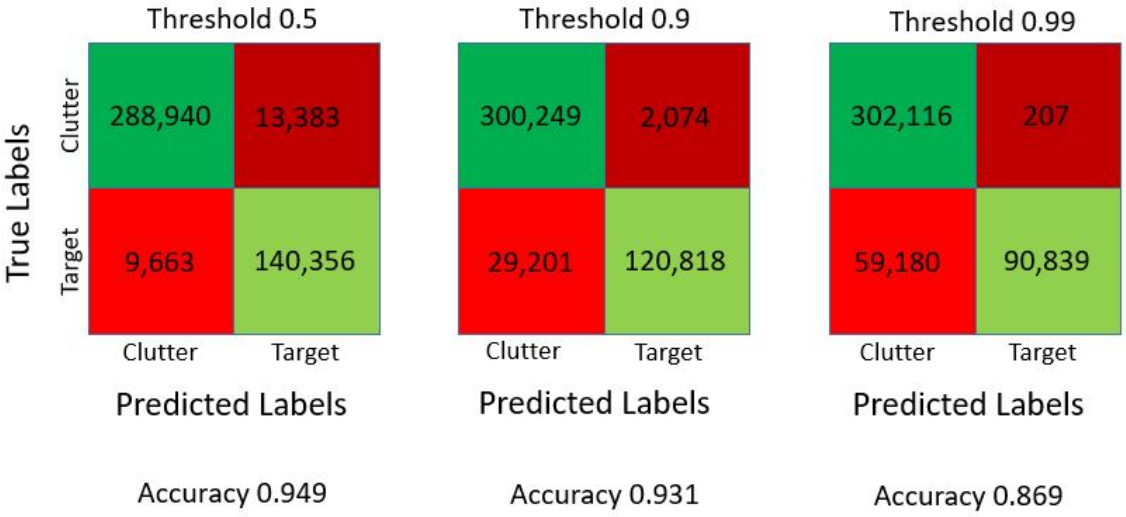


Figure 4.4: Confusion matrices based on results with model trained on data from Mode 3. Three examples of thresholds are shown as well as the classification accuracy for each threshold configuration.

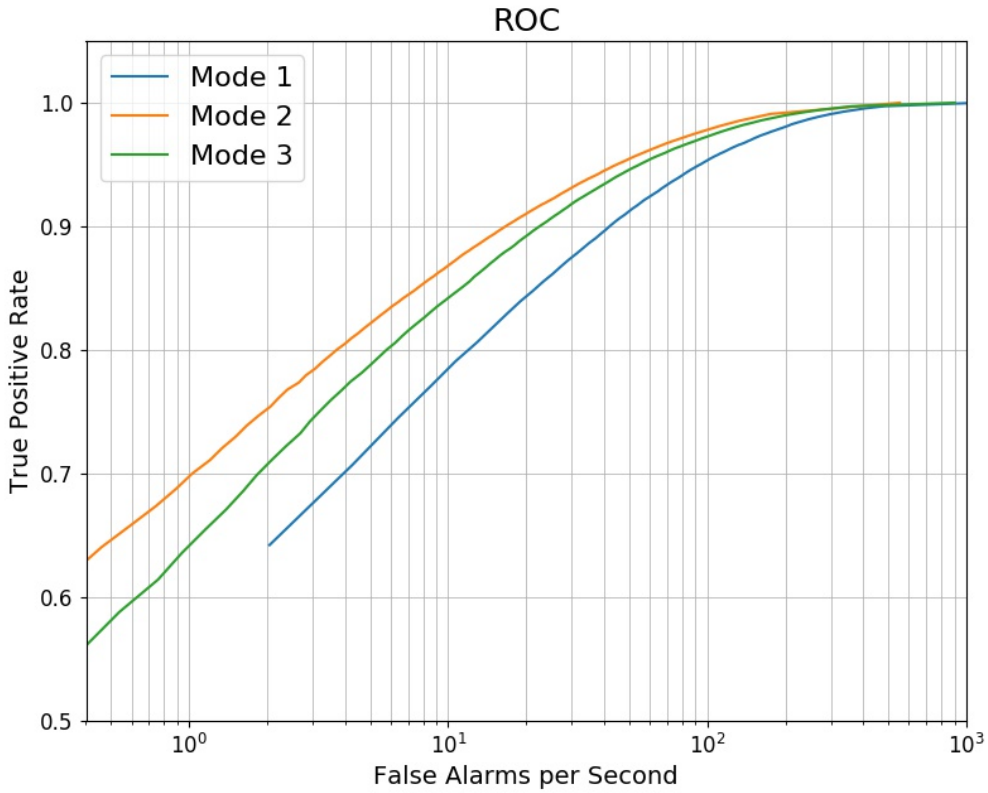


Figure 4.5: ROC curves for three different models trained on different modes.

5
Conclusion

This project has managed to achieve the objective of performing a study of whether machine learning can be used when categorizing small boats from sea clutter based on radar data. Regarding the aims of this study it has been found that CNNs are suitable to solve the image problem established. The best performing model in this study consists of five layers and 33,362 parameters and is shown in Figure 4.3.

The number of epochs used to train each model throughout this study is not presented. This is a choice made since the algorithm developed is intended to be used offline where the time to train a model is irrelevant. However, some choices made in this study has taken the complexity of the model into consideration. This has been made in order to decrease the training time and improve the efficiency in the process of obtaining a best model and is not considered as having any impact on the result.

The overall finding from network architecture and hyperparameter tuning, shown in Section 4.2.1, is that the performance does not rely on these settings too much. This is also shown in 4.2.2 where the increased depth and structure of the residual network, which is state of the art, do not outperform the best performing model of this project. One conclusion is that the patches extracted are relatively small, thus are bounded to extract a certain amount of features and increased depth is thereby not giving a better result. The data parameters, shown in Section 4.3, have a greater impact on the performance of the model. When looking further at the data parameter settings it is concluded that

- The size of the patches does not seem to perform significantly better when enlarging the size above 7x7. Below this patch size, the classification accuracy decreases the smaller the size gets.

- The classification accuracy increases the larger the training data size gets. With this in mind, it is concluded that it is better to use as much training data as possible. Something to take into consideration when enlarging the training data size is though that the number of calculations increases, making it computationally harder and leading to a longer time to train the model.

- Based on the masking operations used, it can be seen that the 3x3 center pixels in a patch give the most information to the classifier. This is good, since the synthetic target model used is not supposed to alter the pixels outside of this

3x3 region too much. However, a little altering is necessary in order for the target model to spread in the range and Doppler domain in a realistic way without creating a sharp edge when added to a background patch. It can also be stated that there is more information in the Doppler domain than in the range domain. This is expected since the small boats of interest in this study will have a higher tendency to spread in the Doppler domain than in the range domain due to the resolution of the radar.

In Section 4.3.1.1, it is shown that the best results are achieved when training a model based on data from the same mode as the test data are extracted from. It is also shown that different clutter levels, as a result of weather condition, make some difference in classification accuracy. It is better to train a model in similar weather conditions as the test data is about to predict. However, since it is hard to predict what the weather conditions will be it is recommended to use data from several flights with the same mode when training in order to obtain a proper spectrum of weather conditions. In Section 4.3.1.2 it is shown that by doing this, and predicting test data from another flight, a high classification accuracy is still obtained. As new flight recording tests are made, data from these flights should also be added to the training data in order to obtain the best generalizing model.

The conclusion from the ROC curve in Section 4.3.1.3 is that it is possible to obtain a manageable level of false alarms per second by adjusting the probability threshold for the classifier to a specific value. Another conclusion drawn from adjusting the threshold is that a trade-off occurs between having a low false alarm rate and having a high classification accuracy. Since this measurement is more commonly used in the radar field, this evaluation method gives another kind of evaluation than the classification accuracy.

5.1 Future Work

In this study, one method of modelling synthetic targets is used. However, this model may be formulated in many ways. For future work it is recommended to consolidate the synthetic target model as well as testing the algorithm with real target data, and not only use a synthetic target model. Even though the latter named model is thought to be advanced enough to make the synthetic targets look similar to real targets, it is hard to make a fair comparison against using real targets.

It is also recommended to implement the developed algorithm to perform classification in real time in order to evaluate the true performance. In this project, the time aspect has not been necessary to take into consideration but if an online version is to be implemented other types of network may be of interest.

This study has viewed the problem of categorizing small boats from sea clutter as a one channel image problem. However, other implementations may be evaluated such as using the complex radar data separately by using different networks for the real and imaginary values.

Bibliography

[1] Findclarissa, "File:Wave Diagram.png." https://commons.wikimedia.org/wiki/File:Wave_Diagram.png, 2016.

[2] Pixabay, "Boat Image." https://pixabay.com/sv/b%C3%A5t-ocean-rekreation-segelb%C3%A5t-2028042/, 2017.

[3] G. Stimson, *Introduction to Airborne Radar*. Aerospace & Radar Systems, SciTech Pub., 1998.

[4] K. Ward, S. Watts, R. Tough, I. of Engineering, Technology, and I. of Electrical Engineers, *Sea Clutter: Scattering, the K Distribution and Radar Performance*. Electromagnetics and Radar Series, Institution of Engineering and Technology, 2006.

[5] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press, 2014.

[6] Stanford, "Optimization: Stochastic Gradient Descent." http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/, 2018.

[7] O. Enqvist, "Lecture Notes in SSY097 Image Analysis at Chalmers University of Technology," 2017.

[8] A. Karpathy, "University Lecture: Convolutional Neural Networks for Visual Recognition." http://cs231n.github.io/convolutional-networks/, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[10] P. Dammert, "Internal report at SAAB Electronic Defense Systems," 2018.

[11] D. Barton, *Modern Radar System Analysis*. Artech House Radar Library, Artech House, 1988.

[12] J. Brownlee, "How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras." https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/, 2016.

[13] F. Chollet *et al.*, "Keras." https://github.com/keras-team/keras, 2015.

[14] S. Lau, "Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning." https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1, 2017.

[15] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

A

Tables of Hyperparameters

This appendix shows the full tables of testing different hyperparameters for the optimizer SGD as well as testing of dropout rate and weight constraints.

A.1 Hyperparameter testing for optimizer SGD

Table A.1: Classification accuracy for different choices of learning rate and momentum rate with SGD as optimizer.

Learning Rate	Momentum	Accuracy
0.0001	0.0	0.883
0.0001	0.2	0.885
0.0001	0.4	0.886
0.0001	0.6	0.888
0.0001	0.8	0.892
0.0001	0.9	0.894
0.001	0.0	0.895
0.001	0.2	0.899
0.001	0.4	0.900
0.001	0.6	0.900
0.001	0.8	0.903
0.001	0.9	0.905
0.01	0.0	0.905
0.01	0.2	0.904
0.01	0.4	0.905
0.01	0.6	0.905
0.01	0.8	0.897
0.01	0.9	0.900
0.1	0.0	0.898
0.1	0.2	0.895
0.1	0.4	0.884
0.1	0.6	0.893
0.1	0.8	0.877
0.1	0.9	0.833

Learning Rate	Momentum	Accuracy
0.2	0.0	0.872
0.2	0.2	0.891
0.2	0.4	0.888
0.2	0.6	0.880
0.2	0.8	0.827
0.2	0.9	0.789
0.3	0.0	0.867
0.3	0.2	0.885
0.3	0.4	0.887
0.3	0.6	0.815
0.3	0.8	0.800
0.3	0.9	0.598

A.2 Dropout Rate and Weight Constraint

Table A.2: Classification accuracy for different choices of dropout rate and weight constraint.

Dropout	Weight Constraint	Accuracy
0.0	1	0.906
0.0	2	0.905
0.0	3	0.906
0.0	4	0.905
0.0	5	0.907
0.1	1	0.902
0.1	2	0.903
0.1	3	0.903
0.1	4	0.903
0.1	5	0.903
0.2	1	0.899
0.2	2	0.903
0.2	3	0.901
0.2	4	0.902
0.2	5	0.903
0.3	1	0.898
0.3	2	0.900
0.3	3	0.900
0.3	4	0.901
0.3	5	0.900
0.4	1	0.897
0.4	2	0.897
0.4	3	0.898
0.4	4	0.898
0.4	5	0.897
0.5	1	0.894
0.5	2	0.896
0.5	3	0.896
0.5	4	0.895
0.5	5	0.897
0.6	1	0.892
0.6	2	0.890
0.6	3	0.894
0.6	4	0.893
0.6	5	0.893
0.7	1	0.887
0.7	2	0.889
0.7	3	0.889
0.7	4	0.889
0.7	5	0.889
0.8	1	0.883
0.8	2	0.885
0.8	3	0.879
0.8	4	0.884
0.8	5	0.881
0.9	1	0.867
0.9	2	0.866
0.9	3	0.870
0.9	4	0.867
0.9	5	0.866