

KTH ROYAL INSTITUTE
OF TECHNOLOGY

Doctoral Thesis in Electrical Engineering

Radar Signal Processing using Artificial Neural Networks

ALEXANDER KARLSSON

Stockholm, Sweden 2023

Radar Signal Processing using Artificial Neural Networks

ALEXANDER KARLSSON

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Friday the 13th October 2023 at 10:00 CET in F3, Lindstedtsvägen 26, Stockholm.

Doctoral Thesis in Electrical Engineering
KTH Royal Institute of Technology
Stockholm, Sweden 2023

© Alexander Karlsson

TRITA-EECS-AVL-2023:61
ISBN 978-91-8040-689-5

Printed by: Universitetsservice US-AB, Sweden 2023

Abstract

This thesis combines radar signal processing, with data driven artificial neural network (ANN) methods. Signal processing algorithms are often based on modeling assumptions of how the data was formed. In some cases, such models are sufficient for designing good, or even optimal, solutions.

In many cases however, these models may be too complicated to form analytical solutions; be too simplified, such that practical results may differ significantly from what was theoretically indicated; be unknown in the sense that one of several known models or parameter values may fit the data, but we do not know which; or be too complex such that the solution will be too heavy to compute.

Data driven ANN methods provide a simple way of bridging these gaps. We demonstrate this in four different studies, where we make use of radar data models to formulate data driven solutions that are both accurate and computationally efficient.

We compare ANN based results to computationally demanding least squares, and exhaustive matched filtering approaches. We show that the performance of the ANNs are comparable to these, but come at a fraction of the computational load. We train ANNs on data sampled from models using a wide range of parameter values. This naturally handles drifts and unknown parameter values, which may change the data, but not the desired prediction. We show that an ANN classifier trained on data from simple models may in practice perform significantly worse than what was expected from theory. We improve this by combining a limited amount of real data with synthetic model data. In all cases, we make use of models that are simple to evaluate. They are however not simple to analyze for the purpose of creating analytical solutions.

In particular we present a method for achieving non-coherent pulse compression that resolves targets within a single pulse width. We present a method for detecting weak target trajectories that does not require prior assumptions on target acceleration, the signal-to-noise ratio, etc. We present different methods of incorporating imperfect model data in the training of classifiers of drone and non-drone targets. Finally we present a method for estimating the path difference in sea surface multipath propagation, for the purpose of target tracking.

Keywords

Artificial neural networks, classification, deep learning, detection, parameter estimation, radar

Sammanfattning

Denna avhandling kombinerar radarsignalbehandling med datadrivna artificiella neuronnät (ANN) metoder. Signalbehandlings algoritmer är ofta baserade på modellantaganden om hur data kommit till. I vissa fall är sådana modeller tillräckliga för att designa bra, eller till och med optimala, lösningar.

I många fall kan dock dessa modeller vara alltför komplicerade för att ta fram analytiska lösningar; vara alltför förenklade, så att resultat från praktik skiljer sig markant från vad som var teoretiskt väntat; vara okända i den bemärkelsen att en av många kända modeller eller parameter värden kan passa data, men vi vet inte vilket; eller vara alltför komplexa så att lösningen blir för tung att beräkna.

Datadrivna ANN metoder kan användas för att brygga dessa gap. Vi demonstrerar detta i fyra olika studier, där vi använder oss av modeller för radardata för att formulera datadrivna lösningar som är både noggranna och beräkningsmässigt effektiva.

Vi jämför resultat från ANN lösningar med resultat från beräkningsmässigt mer krävande minsta kvadrat och matchade filter lösningar. Vi visar att prestandan av ANN är jämförbar med dessa, men kräver endast en bråkdel av beräkningarna. Vi tränar ANN på data från modeller med ett brett span av parameter värden. Detta för att hantera okända parameter värden och förändringar i dessa, som kan ändra data men inte den önskade prediktionen. Vi visar att en ANN klassificerare som är tränad på data från enkla modeller kan prestera anmärkningsvärt sämre i praktiken från vad som vad var teoretiskt väntat. Vi förbättrar detta genom att kombinera en begränsad mängd riktigt data med syntetiskt modelldata. I samtliga fall använder vi oss av modeller som är enkla att exekvera. Dom är dock inte enkla att analysera i avsikt att skapa analytiska lösningar.

Mer konkret presenterar vi en metod för icke-koherent pulskompression som löser upp mål inom en pulsvidd. Vi presenterar en metod för att detektera svaga målspår som inte kräver antaganden om målets acceleration, signal-till-brus förhållande etc. Vi presenterar olika metoder för att inkorporera bristande modelldata i träningen av klassificerare för drönare och icke-drönare mål. Till sist presenterar vi en metod för estimering av skillnaden i sträcka under flervägsutbredning över sjö, i avsikt av målföljning.

Nyckelord

Artificiella neuronnät, klassificering, djupinlärning, detektion, parameter estimering, radar

Acknowledgments

I would first and foremost like to thank my wife Hanna for her great support. I extend my thanks to our children Gunnar, Folke, Sigvard and Teolina for their patience. This thesis could not have been realised without the guidance from my supervisors Magnus Jansson at KTH and Mikael Hämäläinen and Henrik Holter at SAAB, as well as the financial and educational support from both the Wallenberg AI, Autonomous Systems and Software Program (WASP), SAAB and KTH. I thank my many colleagues at SAAB, KTH and those within WASP for fun and interesting discussions which have been of great help and joy. I also thank Saikat Chatterjee for acting as the advance reviewer of this thesis and Mats Bengtsson for acting as co-supervisor. I thank the Digital Image Museum Associates in Florence and the Los Angeles County Museum of Art for the permission to use the image “La trahison des images”.

My parents Carl-Axel and Katarina, as well as my family as a whole, are due much credit. I would not have been able to write this thesis, had it not been for the support and encouragement I have received from them in the course of my days. Finally I would like to extend my appreciation to all the people who answer Latex and Matlab related questions on various forums on the internet. This record of my proceedings in my research would not have been the same without them.

Stockholm, August 2023
Alexander Karlsson

Contents

List of Papers **xi**

List of Figures **xiii**

List of Tables **xvii**

List of Acronyms **xix**

List of Notations **xxiii**

1 Introduction **1**
1.1 Structure of the Thesis . . . 3
1.2 Notations . . . 4
1.3 Machine Learning and Signal Processing . . . 4
1.4 Contributions . . . 7

2 Background in Radar Signal Processing **9**
2.1 General Background in Radar . . . 9
2.1.1 A Brief History of Radar . . . 9
2.1.2 Applications . . . 14
2.1.3 Radar Frequency Bands . . . 16
2.2 Basic Principles . . . 18
2.2.1 The Radar Equation . . . 20
2.2.2 The Signal Processing Chain . . . 22
2.3 Matched Filtering . . . 24
2.4 Doppler Processing . . . 25
2.5 Antennas . . . 30
2.5.1 Phased Array Antennas . . . 32
2.5.2 Monopulse . . . 36
2.6 Detection . . . 38

2.7 Target Modeling and the Radar Cross Section 41
2.8 Sidelobes and Windowing 47
2.9 Frequency Modulated Continuous Wave 48

3 Background in Artificial Neural Networks 53
3.1 Artificial Neural Networks 54
3.1.1 Universal Function Approximators 56
3.2 Architectures 58
3.2.1 Convolutional Neural Networks 58
3.2.2 Autoencoders with Artificial Neural Networks 62
3.3 Training Artificial Neural Networks 64
3.3.1 Loss Functions 64
3.3.2 Backpropagation 66
3.3.3 Stochastic Gradient Descent 69
3.3.4 Adaptive Moment Estimation 69
3.3.5 Parameter Initialisation 71
3.3.6 Training, Validation and Testing Data 72
3.3.7 Network Pruning 76
3.3.8 Understanding the Function 77

4 Stepped Frequency Pulse Compression with Non-Coherent Radar 81
4.1 Introduction 81
4.2 System Model 83
4.2.1 Signal model 83
4.2.2 The Label 86
4.2.3 Ambiguities and the SNR 88
4.2.4 Network model 90
4.3 Results 92
4.3.1 Training the CNN 93
4.3.2 Resolution Gain 96
4.3.3 SNR Gain 100
4.3.4 Other Scenarios and Methods 105
4.4 Conclusions 109

5 Data Driven Track Before Detect 111
5.1 Introduction 111
5.2 Data Model 113
5.3 Detection Method 115
5.4 Results 118
5.5 Conclusions 126

6 Drone Classification **127**
6.1 Introduction . . . 127
6.2 Data and Preprocessing . . . 130
6.2.1 Real Data . . . 130
6.2.2 Synthetic Data . . . 133
6.3 Approach One: Model-Aided Classification . . . 137
6.3.1 Classifier Model . . . 137
6.3.2 Results . . . 138
6.4 Approach Two: Anomaly Based Classification . . . 143
6.4.1 Classifier Model . . . 143
6.4.2 Results . . . 146
6.5 Conclusions . . . 150

7 Low-Angle Target Tracking in Sea Surface Multipath **153**
7.1 Introduction . . . 153
7.2 System Model . . . 157
7.2.1 Geometric Model . . . 157
7.2.2 Surface Reflection Model . . . 160
7.2.3 Radar Model . . . 161
7.2.4 Signal Model . . . 162
7.3 Parameter Estimation . . . 167
7.3.1 Height Estimation . . . 167
7.3.2 Maximum Likelihood and Least Squares . . . 168
7.3.2.1 ML on $\eta(\tilde{f})$ . . . 168
7.3.2.2 LS on $\eta(\tilde{f})$ . . . 170
7.3.2.3 LS on $|\Sigma_x(\tilde{f})|$ . . . 170
7.3.3 Convolutional Neural Network . . . 172
7.3.3.1 Data Preprocessing . . . 172
7.3.3.2 Data Sampling . . . 172
7.3.3.3 Labels . . . 173
7.3.3.4 Network Architecture . . . 176
7.3.3.5 Training the CNN . . . 176
7.3.4 Tracking and Filtering Estimates . . . 178
7.3.4.1 The Kalman Filter and Target Dynamics . . . 180
7.4 Results . . . 183
7.4.1 Comparison with ML and LS methods . . . 183
7.4.2 Filtering the Estimate of n . . . 185
7.4.3 Normal Tracking and Multipath Tracking . . . 187
7.4.4 Comparison with Direction of Arrival Methods . . . 189
7.4.5 Alternative Models . . . 191

7.4.5.1 Clutter and Noise Models 192
7.4.5.2 The value of N_f 192
7.4.5.3 Reflection Coefficient Models 193
7.4.5.4 The Effect of the Refraction Factor k 196
7.5 Conclusions 198

8 Conclusions and Future Work **201**
8.1 Conclusions 201
8.2 Future work 204
8.3 Reflections 204

References **207**

List of Papers

This thesis is based on material from the following papers:

A **Stepped Frequency Pulse Compression With Noncoherent Radar Using Deep Learning**, Alexander Karlsson, Magnus Jansson, Henrik Holter, *IEEE Transactions on Aerospace and Electronic Systems, vol. 57, no. 3, pp. 1657-1671, June 2021, doi: 10.1109/TAES.2020.3046336*

B **Model-Aided Drone Classification Using Convolutional Neural Networks**, Alexander Karlsson, Magnus Jansson, Mikael Hämäläinen, *2022 IEEE Radar Conference (RadarConf22), New York City, NY, USA, 2022, pp. 1-6, doi: 10.1109/RadarConf2248738.2022.9764194*

C **Low-Angle Target Tracking in Sea Surface Multipath Using Convolutional Neural Networks**, Alexander Karlsson, Magnus Jansson, Mikael Hämäläinen, *IEEE Transactions on Aerospace and Electronic Systems, doi: 10.1109/TAES.2023.3282191, June 2023 (Early Access)*

D **Anomaly Based Drone Classification Using a Model Trained Convolutional Neural Network Autoencoder on Radar Micro-Doppler**, Alexander Karlsson, Magnus Jansson, Mikael Hämäläinen, *IEEE International Radar Conference 6-10 Novemebr 2023, Sidney Australia.*

E **Data Driven Track Before Detect Using Artificial Neural Networks**, Alexander Karlsson, Magnus Jansson, Mikael Hämäläinen, *IEEE International Radar Conference 2023 6-10 Novemebr 2023, Sidney Australia.*

List of Figures

1.1 Analytical signal processing and data driven machine learning. . . 5

2.1 Examples of an A and B scope/display... 19
2.2 Example of the data cube... 23
2.3 Examples of coherent (a) and non-coherent (b) pulse compression... 26
2.4 Example of Doppler Processing... 29
2.5 FMCW radar measurement of a drone... 29
2.6 The dipole antenna... 31
2.7 ULA structure... 33
2.8 The far field antenna array directivity using dipole elements... . . . 34
2.9 Example of an analog ESA (a), and a digital ESA (b)... 35
2.10 Different monopulse configurations... 37
2.11 Angle error estimate for both amplitude (A) and... 37
2.12 Example of a CFAR window. 39
2.13 CFAR Example with three targets... 40
2.14 A fishing fly imitation of a wasp... 41
2.15 Mie scattering of a conducting sphere... 43
2.16 RCS in dBsm as a function of aspect angle... 45
2.17 The RCS pattern of a cylinder... 46
2.18 Example of a Taylor window on a 32 element ULA... 48
2.19 A two target FMCW example... 49
2.20 The Saab SIRS 1600 FMCW radar. 52

3.1 Example of nonlinearity in binary classification... 55
3.2 Illustration of a three layer MLP... 57
3.3 Illustration of a two layer CNN architecture... 59
3.4 Example of an autoencoder... 63
3.5 Examples of gradient descent... 71
3.6 Examples of different training and testing sets... 73

3.7 Training and test results for the three training datasets in Fig. 3.6. . 75
3.8 Activations of the MLP in Table 3.4... 76
3.9 “La trahison des images” (The Treachery of Images)... 77
3.10 Examples of equivalent filters... 80

4.1 Training and validation loss for different sizes of training data... . . 94
4.2 Example of improvement with SNR for a five target scenario... . . 95
4.3 Numerically evaluated synthetic pulse shape... 97
4.4 Example of peak-to-valley ratios... 98
4.5 Empirical median peak-to-valley ratio... 99
4.6 Example of a simulated scan with three targets... 101
4.7 Empirical mean gain in SNR as a function of SNR... 102
4.8 Example of compressed pulse for different SNRs... 103
4.9 ROC curves for different scenarios... 104
4.10 Comparison between the iterative GLA solution and... 106
4.11 Scenario with two moving targets... 107
4.12 Scenario with a rotating target... 108

5.1 Examples of trajectories... 114
5.2 Empirical ROC curves for different SNR values... 119
5.3 Empirical detection probability as a function of SNR... 120
5.4 The RMSE in range cells... 121
5.5 RT-frame example with weak targets. 123
5.6 RT-frame example with strong and weak targets. 124
5.7 Example of a real radar measurement. 124
5.8 Example with a larger RT-frame. 125

6.1 Illustration of the considered signal processing chain... 129
6.2 Distributions of estimated RCS values at 77 GHz... 131
6.3 Example of parameter selection in the drone model. 134
6.4 Examples of the micro-Doppler spectra... 136
6.5 Confusion matrix for the nine classes... 139
6.6 Mean classification accuracy as a function of SNR... 140
6.7 Classification accuracy for the nine classes... 141
6.8 Mean of all equivalent filters... 142
6.9 Projection of the latent space Z... 146
6.10 Mean classification accuracy for drone and non-drone classes... . . 148
6.11 ROC curves where a true positive is a drone classified as an anomaly...149

7.1 Illustration of a low angle scenario... 154
7.2 Illustration of the two target model... 155

7.3 Geometry of the scenario in Fig.7.1... 157
7.4 Multipath interference pattern... 159
7.5 The real and imaginary part of the angle error ε_{θ_1}... 166
7.6 The RMSE of $\hat{\theta}_1^{\text{MP}}$ when $|\delta_1| < T_{\delta_1}$... 175
7.7 Block diagram of the CNN and tracking filter... 182
7.8 RMSE of h_t as a function of the SNR... 185
7.9 Estimate of $\Delta\nu$ for different values of k... 186
7.10 Estimate of the initial n from $\hat{\boldsymbol{y}}_2$ and $\hat{k}$... 186
7.11 Example of recovery from incorrect estimates of n... 187
7.12 Empirical receiver operating characteristic... 188
7.13 Filtered CNN estimates for the two benchmark scenarios... 189
7.14 MSE of h_t for case 1... 191
7.15 RMSE of h_t in the test set for different noise/clutter models... . . . 193
7.16 RMSE of h_t as a function of the number of pulses per batch... . . 194
7.17 Accuracy on n and RMSE of h_t when $\hat{n} = n$... 194
7.18 Case 2 in Table 7.6 with a different reflection model... 195
7.19 RMSE of the CNN estimate of the normalized path difference... . 196
7.20 The upper bound in magnitude of ε_{h_t}... 197

List of Tables

2.1 IEEE Radar Frequency Bands [39] 17
2.2 FMCW Radar Specifications 52

3.1 Fully Connected and Convolutional Networks 61
3.2 Autoencoder using ANNs 62
3.3 Default Parameter Values for Adam 70
3.4 MLP for RGB Colors 74

4.1 Parameter Values 87
4.2 Neural Network Architecture 92
4.3 Batch Size and Loss 93
4.4 Number of Pulses M and Loss 93
4.5 Two Target Required Range† Separation... 100
4.6 Multiple Target Required Range† Separation... 100

5.1 Parameters for Data Generation 115
5.2 Neural Network Architecture 116
5.3 Empirical False Track Count for $T_2 = 3$ 122

6.1 Datasets, available at [143]. 132
6.2 Model Parameters for Synthetic Data 137
6.3 Neural Network Architecture 138
6.4 Model Parameters for Synthetic Data 143
6.5 Neural Network Architecture 144
6.6 Latent Space Dimensionality Results from 20 runs 147
6.7 Classification Accuracy, SNR=20 dB, M=10 150

7.1 Pulse Repetition Frequency 161
7.2 Parameters for Data Generation 173

7.3 Neural Network Architecture 177
7.4 Batch Execution Time 184
7.5 Filter Parameters 188
7.6 Scenario Parameters 188

List of Acronyms

Adam	adaptive moment estimation
ADC	analog to digital converter
AESA	active electronically scanned array
AI	artificial intelligence
ANN	artificial neural network
ATR	automatic target recognition
AUC	area under receiver operating characteristic curve
AWGN	additive white Gaussian noise
CA-CFAR	cell averaging CFAR
CEROS	CelsiusTech radar and optronic sensors
CFAR	constant false alarm rate
CHASE	CelsiusTech high accuracy sea-skimmer estimator
CNN	convolutional neural network
CPI	coherent processing interval
CRB	Cramér Rao bound
CUT	cell under test
CW	continous wave
dBi	decibels relative to an isotropic antenna
dBsm	decibels relative to one square meter
DFT	discrete Fourier transform
DNN	deep neural network
DOA	direction of arrival
ESA	electronically scanned array
ESPRIT	estimation of signal parameters via rotational invariance techniques

FFT	fast Fourier transform
FMCW	frequency modulated continuous wave
GLA	Griffin-Lim algorithm
GPU	graphics processing unit
HF	high frequency
IDFT	inverse discrete Fourier transform
IEEE	Institute of Electrical and Electronics Engineers
IF	intermediate frequency
LNA	low-noise amplifier
LS	least squares
MFD	multi frame detection
ML	maximum likelihood
MLP	multilayer perceptron
mmWR	millimeter-wave radar
MNIST	Modified National Institute of Standards and Technology
MODE	method of direction estimation
MSE	mean squared error
MUSIC	multiple signal classification
OOK	on-off keying
OS-CFAR	ordered statistic CFAR
PCA	principal component analysis
PCL	passive coherent location
PDF	probability density function
PESA	passive electronically scanned array
PRF	pulse repetition frequency
PRI	pulse repetition interval
QDA	quadratic discriminant analysis
RADAR	radio detection and ranging
RCS	radar cross section

ReLu	rectified linear unit
RGB	red green blue
RMS	root mean square
RMSE	root mean square error
RMSProp	root mean square propagation
RNN	recurrent neural network
ROC	receiver operating characteristic
RPS	rounds per second
RT	range-time
SAR	synthetic aperture radar
SCNR	signal-to-clutter-plus-noise ratio
SCR	signal-to-clutter ratio
SINR	signal-to-interference-plus-noise ratio
SNR	signal-to-noise ratio
SO-CFAR	smallest of CFAR
STFT	short time Fourier transform
TBD	track before detect
TWT	traveling-wave tube
UAV	unmanned aerial vehicle
UHF	ultra high frequency
ULA	uniform linear array
VHF	very high frequency
WASP	Wallenberg AI, Autonomous Systems and Software Program
WSF	weighted subspace fitting
WWII	world war two

List of Notations

$\mathbb{R}$	set of real numbers
$\mathbb{C}$	set of complex numbers
a	scalar or scalar valued function
$\boldsymbol{a}$	vector or vector valued function
$\boldsymbol{A}$	matrix
$\circ$	Hadamard product
$\boldsymbol{A}[i,j]$	element of a matrix where $\boldsymbol{A}[i,j] = a_{i,j}$
$g(\cdot)$	a function of some parameter(s)
$\hat{x}$	an estimate of a parameter x
$\bar{x}$	a filtered or in some way averaged value of x
$\|\|\boldsymbol{x}\|\|_2$	the Euclidean norm of a vector $\boldsymbol{x}$
$\boldsymbol{x} \cdot \boldsymbol{y}$	dot product or scalar product between $\boldsymbol{x}$ and $\boldsymbol{y}$
$\|x\|$	absolute value or modulus of (complex valued) x
$\angle x$	the argument or angle of complex valued x
x^*	complex conjugate of x
$\boldsymbol{x}^T$	transpose of a vector $\boldsymbol{x}$
diag($\boldsymbol{x}$)	a diagonal matrix where the diagonal elements are given by the elements in $\boldsymbol{x}$
diag($\boldsymbol{X}$)	the diagonal elements of matrix $\boldsymbol{X}$
$\boldsymbol{I}$	identity matrix

$\forall$	for all
$\propto$	proportional to
$\propto\!\!\!\!\sim$	approximatley proportional to
$x \sim \mathcal{U}(\cdot)$	x drawn from a distribution $\mathcal{U}(\cdot)$
$\mathcal{U}(a, b)$	uniform distribution with limits a and b
$\mathcal{CN}(\mu, \sigma^2)$	complex normal/Gaussian distribution with mean μ and variance σ^2
$\mathcal{F}\{g(t)\}$	the Fourier transform of $g(t)$
$\mathbb{E}[x]$	Expected value of random variable x
$[a, b]$	an interval between and including a and b
$[a, b)$	an interval between a and b, including a but not b
$x \in [a, b]$	x belongs to the interval $[a, b]$
$\lfloor \cdot \rfloor$	floor function
$\lceil \cdot \rceil$	ceiling function
c	Speed of light, 299792458 m/s
k_{B}	Boltzmann's constant, 1.38×10^{-23} W/(HzK)

Chapter 1

Introduction

The process of extracting information from signal data is an apt description of signal processing. Finding good methods, or algorithms, that accomplish this, is a fundamental research topic and constitutes the basis of this thesis. Specifically, we look at signal processing in radar using artificial neural networks (ANNs), a special type of machine learning method in the field of artificial intelligence (AI).

Machine learning is a collective term for methods in which algorithm or function parameters are determined by data i.e., they are data driven. A proper understanding of data driven machine learning methods require an understanding of both the algorithms used, and the data these are trained and evaluated on. The latter is often referred to as domain knowledge. That being said, machine learning algorithms are often advocated as methods that do not require domain knowledge and can be used on a diverse set of problems with a similar approach. Although there is truth in this, we will in this thesis highlight the benefits of using domain knowledge in radar, with ANNs.

We first note that the achievable results are not only limited by the complexity of a particular ANN structure, but are also limited by the data itself. As an example, for sufficiently poor signal-to-noise ratio (SNR) values, it will be impossible to distinguish sensor data from random noise, regardless of the method we use. This type of noise is typically not encountered in the field of computer vision and natural language processing, from which much of the current ANN developments have come. The effect of noise in radar however, as well as in wireless communication in general, is central to most signal processing methods.

Another contrast to computer vision tasks, is our understanding of radar data and our ability to model it. This is aided by decades of research in signal processing and radar, and has a rich literature, especially when compared to the modeling of

images of objects in computer vision.

Our designs and motivations for the various ANN structures are all motivated by the nature of the data, or the model descriptions of the data. However due to a limited analytical understanding of ANNs, see Chapter 3, we cannot make any claims on optimally regarding the network architectures. We can however extensively evaluate the solution and compare the results to theoretical bounds from the signal processing domain, such as the Cramér Rao bound used in Chapter 7.

In some cases a model of the data may be sufficient for designing the optimal algorithm for some specific purpose. In many cases however, these models may be too complicated to form analytical solutions; be too simplified, such that practical results may differ significantly from what was theoretically expected; be unknown in the sense that one of several known models or parameter values may fit the data, but we do not know which; or be too complex, such that the solution will be too heavy to compute.

Data driven ANN methods provide a simple way of bridging these gaps. We demonstrate this in four different studies, aimed at specific radar signal processing tasks. In these studies we address the following problems.

Models may be too complex, such that the solution will be too heavy to compute. Whenever models of data exists, it is always possible to form a least squares solution to the whatever optimisation problem is at hand, see Section 1.3. For non-convex problems, this may involve testing a large amount of combinations of model parameter values. We compare an ANN solutions to such a least squares and maximum likelihood search, when estimating the path difference in multipath propagation. We also compare another ANN solution with an extensive matched filter solution for detecting weak target trajectories. We show that the performance of these ANNs are comparable to the in some sense optimal solutions, but come at a fraction of the computational load. By combining modeling assumptions with data driven ANN based approaches, we obtain solutions that are faster to execute than what would otherwise be possible. This enables fast and accurate real time processing of streaming senor data.

Models may be unknown in the sense that one of several known models or parameter values may fit the data, but we do not know which. When we have accurate models of the data, we may still need good parameter values in order to formulate a heuristic, or even an optimal, solution to the problem. In these cases we may assume that the true parameter values lie within some interval. In our data driven methods we then sample parameters within these limits, from which we generate data using a model. We show in all four studies that an ANN trained on this kind of model data, interpolates well between different data points. We use this to achieve the following.

- We map different signal patterns to different range profiles of unresolved

targets in the application of non-coherent pulse compression. This works independently of the particular target fluctuation model, or scattering model.

- We detect target trajectories with weak SNR values, without prior specification of any parameters such as the target acceleration value, SNR etc.
- We distinguish drones from birds using model data, but do not require specific assumptions on parameter values such as propeller size, number of propellers, rotational speed etc.
- Finally, we estimate the path differences in sea surface multipath propagation for the purpose of target tracking, and do so independently of the particular parameter value of the reflection coefficient of the sea surface.

Models may be too simplified, such that practical results may differ significantly from what was theoretically expected. In the area of automatic target recognition, ANNs may be used to distinguish between different classes of targets. We use simplified models of the signal pattern from drones, and show that ANN classifiers that are trained on data from such models, perform significantly worse than what is obtained when training an ANN on real data. Having real data on all possible instances is however difficult to realise in practice, and we therefor combine a limited amount of real data with simulated data. We present two different methods of how to incorporate imperfect model data with the training of ANN based classifiers. By so doing, we significantly improve the classification results.

Models may be too complicated to form analytical solutions. In all cases we make use of models that are simple to execute. These will however often give little, or insufficient analytic intuition of how to form a desired solution analytically. The exception being the trivial least squares approach discussed in Section 1.3. Although it is often possible to make model simplifications in order to obtain manageable solutions, these simplifications may also increase the error between the actual data and the model. In some cases this will lead to relatively poor solutions, such as a particular least squares solution discussed in Chapter 7. ANN based solutions on the other hand, only require an example dataset that is sufficiently diverse to represent the desired function or distribution. This enables the use of even the most complex of signal models, as it only requires the ability to sample from them. As mentioned, it also becomes possible to mix real and simulated data to compensate for differences between model and reality.

1.1 Structure of the Thesis

The rest of this thesis is structured as follows. In Section 1.2 we list some general notations. The machine learning approach used in all the presented studies is

illustrated in Section 1.3. Here we also compare this approach with standard machine learning using ANNs, and a more analytical signal processing approach. The thesis contributions are summarized in Section 1.4. In Chapter 2 we give background to radar signal processing and in Chapter 3 we give background to ANNs. In both these chapters we highlight aspects that are relevant to this thesis. In Chapter 4, we present a study on non-coherent pulse compression. In Chapter 5, we present a data driven method for detecting target trajectories with weak SNRs. In Chapter 6, we present different methods for robust classification of drone and none-drone targets, based on micro-Doppler radar. In Chapter 7, we present a method for tracking targets in sea surface multipath. Finally we give conclusions directions for future work in Chapter 8.

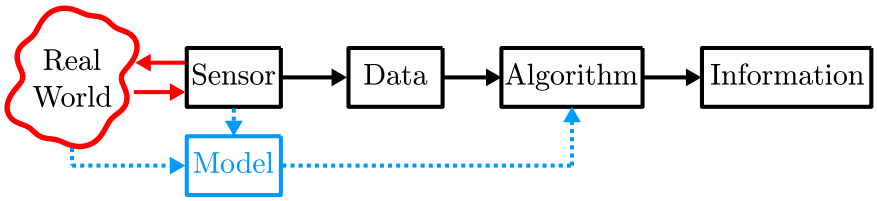
1.2 Notations

All acronyms are listed on page xix. We will use the following general notations. The set of all real numbers is denoted as $\mathbb{R}$, and the set of all complex numbers is denoted as $\mathbb{C}$. Scalars are given in lower case font a, vectors in bold font $\boldsymbol{a}$, and matrices and higher order tensors in upper case bold $\boldsymbol{A}$. A real valued matrix $\boldsymbol{A} \in \mathbb{R}^{N \times M}$, has N rows and M columns. Integers written in upper case normal font such as N and M, are often associated with index values n and m, where e.g. $n = 1, 2, ..., N$. An element of a matrix is thereby given as $\boldsymbol{A}[n, m] = a_{n,m}$ and likewise for vectors $\boldsymbol{a}[n] = a_n$. For the parameters of a scalar valued function g, we use notations g, $g(x)$, $g(x, y)$ etc. interchangeably depending on context. We may also use $g(\cdot)$ to signify that g is a function of some parameter(s), and likewise for vector valued functions, e.g. $\boldsymbol{g}(\cdot)$. An estimate of a data point is denoted with a hat, $\hat{\boldsymbol{x}}$, and an averaged or in some way filtered data point is denoted with a bar, $\bar{\boldsymbol{x}}$. These notations are listed on page xxiii-xxiv, along with other notations introduced throughout this thesis.

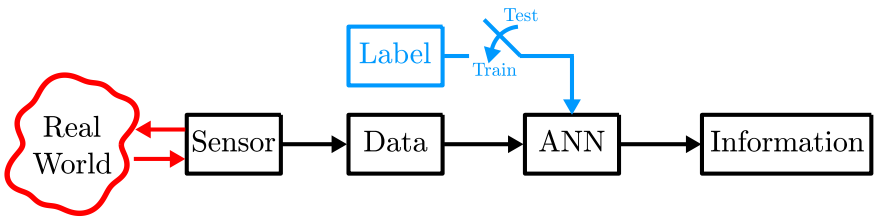
1.3 Machine Learning and Signal Processing

We here point out the different approaches of recent machine learning methods and more analytical signal processing approaches. If we restrict "data" to mean signal data, then machine learning can also be regarded as signal processing, at least from the perspective of extracting information from data.

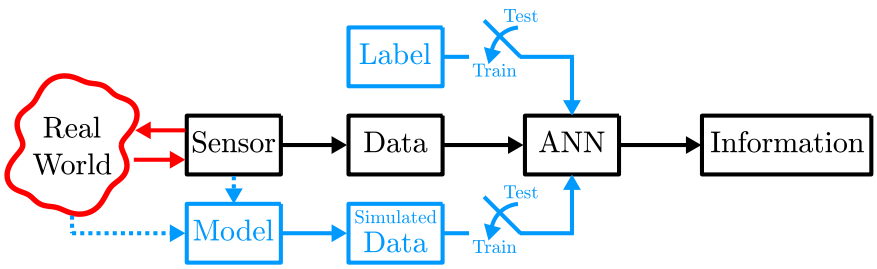
Let $\boldsymbol{x}$ be observed data from which we wish to estimate y. This could e.g. be a particular property/parameter, or a class to which the data belongs. A basic approach for extracting this information is illustrated in Fig 1.1a. Here, data is processed by some algorithm to produce information about the "real world" from



(a) Illustration of a basic signal processing scheme.



(b) Illustration of a supervised learning scheme using an ANN.



(c) Illustration of a supervised learning scheme using an ANN and model data.

Figure 1.1: Analytical signal processing and data driven machine learning.

which the data is obtained. In the cases considered in this theses, the data is obtained via a radar sensor. The algorithm is formulated using a model that takes aspects of both the environment and the sensor into account. This algorithm can take different forms. In the case of parameter estimation, common approaches are

least squares and maximum likelihood. Let our model of the data be

$$\hat{\boldsymbol{x}} = \boldsymbol{g}(\hat{y}) \tag{1.1}$$

where $\boldsymbol{g}(\cdot)$ is a function derived from our domain knowledge of the sensor and the environment, and y is the parameter we wish to estimate. Using least squares, we estimate y as

$$\underset{\hat{y}}{\text{minimize}} \;\; ||\boldsymbol{x} - \boldsymbol{g}(\hat{y})||_2^2 = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2 \tag{1.2}$$

where $||\cdot||_2$ is the Euclidean norm. In some cases, such as linear least squares, the solution can be obtained directly. Other cases may require iterative methods, such as in convex optimisation problems, and sometimes a full grid search is necessary where several feasible values of $\hat{y}$ are tested, perhaps in combination with local gradient search methods. A grid search becomes more challenging when the model also depends on other parameters, as this may require a multidimensional search over all the unknown parameters, not just the the parameter(s) of interest.

In statistical signal processing it is often, but not always, possible to form the likelihood of the observed data given some observation y and possibly also a set of other parameters $\boldsymbol{\theta}$. We write this as

$$p(\boldsymbol{x}|y, \boldsymbol{\theta}). \tag{1.3}$$

where $p(\cdot)$ is the probability density function (PDF) of $\boldsymbol{x}$ given y and $\boldsymbol{\theta}$. The maximum likelihood optimisation problem is then formulated as

$$\underset{\hat{y}, \hat{\boldsymbol{\theta}}}{\text{maximize}} \;\; p(\boldsymbol{x}|\hat{y}, \hat{\boldsymbol{\theta}}) \tag{1.4}$$

and like least squares, is not always simple to solve. For some spacial cases, such as when zero mean Gaussian noise is added to an otherwise deterministic model, the maximum likelihood problem can be formulated as a least squares problem.

A data driven approach using an ANN is illustrated in Fig. 1.1b. During a training phase, the ANN is optimized to map the data $\boldsymbol{x}$ to a label, or ground truth, y, through a large set of examples. We look at this in more detail in Section 3.3. Once the training phase is complete, the ANN operates as any other deterministic function, or

$$\hat{y} = q(\boldsymbol{x}) \tag{1.5}$$

where the function $q(\cdot)$ denotes the ANN.

The labeled data will often come from manual labor, and is not always trivial

to obtain, especially when considering all the factors that determine the data. The fact that some of these factors may be unknown contributes to the "black box" description of ANNs, see Section 3.3.8. The set of training examples therefore needs to be sufficiently diverse, such that the learnt function $q(\cdot)$, behaves as desired for all conceivable data points, not just the ones in the training set. This typically entails very large sets of data. Regularisation methods may also be used, resulting in a functions that generalise better, See Section 3.3. Choosing these regularisation methods are however often associated with much trial and error.

In this thesis, we address this problem with a hybrid approach were we combine the concepts in Fig. 1.1a and Fig. 1.1b. This is illustrated in Fig. 1.1c. Here we use the model in (1.1) to generate a large amount of synthetic data, while simultaneously obtaining labels y for each synthetic $\boldsymbol{x}$. Training an ANN on this data, is a relatively simple method of estimating an inverse function of the model. The performance of this function will depend on the architecture of the ANN and the nature of the data/model.

In Chapter 6 we combine real and simulated data in the training process. In Chapter 5 we train only on simulated data, and evaluate on a limited set of real data. In Chapters 4 and 7, we only train and evaluate on simulated data due to lack of real sensor data. In these cases we test the trained ANNs on data generated from slightly different models in order to simulate differences between real data and model data.

The primary assumption in all studies, is that different situations produce different signal patterns and that an ANN can be trained to map these different patterns to some defining attribute(which is the information sought after). A secondary assumption is that the models we have, are accurate enough to represent the patterns that may arise. Even if this second assumption may fail, one could always produce another model that would be a better match. However, without actual measurements such "improvements" in the model, or flaws in existing models, will be speculative in nature. We therefore limit our studies to models and data that we have at our disposal and acknowledge that further research is warranted, see Chapter 8. Moreover, the use of ANNs is also associated with many other drawbacks and trade-offs. We discuss some of these in Chapter 3.

1.4 Contributions

The contributions of this thesis are linked to using ANNs in different parts of the radar signal processing chain. These are pulse compression, target detection, target classification and target tracking. The contributions can be summarised as follows.

1. We present a method for non-coherent pulse compression. The presented

solution focuses on resolving targets within the pulse length, and differs in this sense from previous non-coherent pulse compression methods using bursts of several pulses for improving the SNR, while maintaining the range resolution. We present these results in Chapter 4, and the results are published in paper A; see page xi.

2. We present a data driven method for detecting target trajectories with weak SNR values, as well as weak trajectories among stronger trajectories. The method is not explicitly based on any assumptions on the targets dynamics. Instead it learns to extract target trajectories from examples in data. We compare these results to a theoretically optimal detector and show that for certain cases the difference between this, and the presented solution, is relatively small. We present these results in Chapter 5, and the results are published in paper E; see page xi.

3. We introduce a special type of regularisation technique for classifying drones and birds with micro-Doppler radar. The regularisation method is based on incorporating synthetic data in the training process. The synthetic data is generated from a model of a drone from the perspective of a radar sensor. We present these results in Chapter 6, and the results are published in paper B; see page xi.

4. Using the micro-Doppler model of drones, we present an anomaly based method for distinguishing drones from non-drone targets. By handling drones as anomalies, we do not need examples of a single real drone in the training process. These results are also presented in Chapter 6, and the results are published in paper D; see page xi.

5. We present an ANN based solution for tracking targets in sea surface multipath. Under certain assumptions, this problem could be approached using least squares or maximum likelihood. We show that the differences in performance between the proposed ANN solution, and the maximum likelihood and least squares solutions are small, whereas the computational time for the ANN is significantly smaller. These results are presented in Chapter 7, and the results are published in paper C; see page xi.

The significance and application of each of these contributions are discussed in greater detail in Chapters 4 - 7.

Chapter 2

Background in Radar Signal Processing

2.1 General Background in Radar

A radar is a devise that detects and locates targets from reflections of electromagnetic waves [1]. All radar systems therefore involve a transmitter of some form, and a receiver. These can be categorised as bistatic radar and monostatic radar. In monostatic radar, the transmitter and receiver are located in the same device. In bistatic radar, the transmitter and receiver are physically separated where the distance between them is on the order of the expected range to the targets [2]. A radar system may also be multistatic where a receiver processes signals from multiple transmitters.

In this thesis, our focus is on the signal processing at the receiving end, primary in monostatic radar. Our discussions on radar are therefore limited to what is necessary for understanding such signal processing, as well as the methods and concepts discussed in Chapters 4-7. In this section we give a brief history of the development of radar and give examples of some typical areas of application. For a more comprehensive account on the fundamentals of radar systems, the reader is referred to works such as [3–7].

2.1.1 A Brief History of Radar

It is not possible to give a full and fair account of the history of radar, nor is it our intention to do so. We will in this section only highlight some of the major milestones in the development of radar and we will give special attention

to technologies that are of particular interest to this thesis. The historical outline presented in this section is in part based on the account in [8].

The concept of radar dates back to the late 19th century, and its development has been closely linked to the progress in radio technology. One of the earliest published experiments that gives hints to what we today call radar, or at least the phenomenon of reflecting electromagnetic waves, was in 1888 by Heinrich Hertz [9, Ch. 8]. To demonstrate that electromagnetic "actions" propagate with finite velocity, he placed a transmitting device near a wall that was covered with a sheet of zinc. Depending on the distance to the wall, the reflected electromagnetic wave would interfere with the oscillating source constructively or destructively. A similar phenomenon can be observed in today's naval radar systems, where reflections from the sea surface interfere with the received signal. This is the underlying problem in our research presented in Chapter 7.

The first operable device to detect radio reflections was patented in 1904 by Christian Hülsmeyer [10]. His "Telemobiloskop" was used to detect ships at ranges of several hundred meters. It did however not measure any time delay or range to the targets. During the subsequent decades, radar technology matured. By the mid 1930s the United states of America, Great Britain, France, Germany, Italy, the USSR (Russia) and Japan had successfully demonstrated radar systems [8]. However, due to their military value, these developments were to the most part held in secret. A possible explanation for this seemingly independent but simultaneous rise of radar technology, can be linked to developments in aviation, where radar came as a natural response to the threat of long range bomber aircraft [8].

Due to limitations in transmit power, many of these early radar systems were of a bistatic continuously transmitting type [8], [11], [12]. See Section 2.2 for comments on transmit power and continuous transmission. A noteworthy example is the Japanese "Type-A" bistatic Doppler interference detector deployed in 1941. Like Hülsmeyer's Telemobiloskop, it offered no range information but detected targets through Doppler shifts when they appeared between the transmitter and receiver stations. The longest such "line" or "barrier" was between Shanghai and the island of Formosa (Taiwan), a distance of 700 km [13].

The first pulsed monostatic radar was developed by the United States Naval Research Laboratory [14]. Tests on an aircraft in December 1934, showed moderate yet promising results. In July 1936, a duplexer was incorporated into the design, allowing a single antenna to be used for both transmission and reception. This finally resulted in the CXAM radar first delivered in the summer of 1940 to the United States Navy. The wavelength of the CXAM radar was 1.5 m (200 MHz) which can be compared to the British pulsed monostatic[1] radar system

[1] Although the system had separate transmit and receive antennas, the distance between these was still negligible when compared to the operational range of the radar. The term semi-monostatic is sometimes used to describe these types of systems.

"Chain Home," with tunable wavelengths between 5.5 and 15 m (20-55 MHz) in the high frequency (HF) band [15]. In fact, most radar systems developed at the time operated at higher frequencies in the very high frequency (VHF) and ultra high frequency (UHF) bands as this was the limit of contemporary vacuum tube technology. Higher frequencies are typically associated with higher spatial resolution. See Section 2.1.3 for comments on radar frequency bands. The in some regards suboptimal choice of frequency range in Chain Home, was justified by its inventor Sir Robert Alexander Watson-Watt, said he "Give them the third best to go on with; the second best comes too late, the best never comes" [16]. The Chain Home was set into 24 hour operation by September 1938 [8], and proved decisive in the outcome of world war two (WWII) beginning only a year later, and in particular in the battle of Britain, almost two years later.

The upper limit in frequency was overcome by the invention of a special type of magnetron, the cavity-resonator magnetron. Invented in 1940 by Henry Boot and John Randall at the university of Birmingham, the first such magnetron produced wavelengths of 9.8 cm (3 GHz) at a considerable output power [17]. Cooperation with the Americans was soon initiated, and at a visit to Bell Telephone Laboratories in October 1940, the magnetron produced an output power of 10 kW. This was five times the output power of the American equipment at over four times the frequency, causing much enthusiasm [17]. By 1944 advancements in the design had increased the output power to 2 MW.

The higher microwave frequencies did not only enable higher resolution in range (due to higher bandwidths), but also allowed smaller antennas with more narrow beamwidths, paving the way for airborne radar. An example is the American Eagle AN/APQ-7 X-band radar, which was put into use by the end of WWII [18]. It operated at 9.4 GHz with 0.4 μs pulse width (or 2.5 MHz bandwidth) and utilised a 16 ft (5 m) mechanically steered phased array antenna yielding a 0.4° beamwidth. This was small enough to be mounted on bomber aircraft.

A significant milestone in radar tracking was the introduction of monopulse, [19]. By comparing the energy from multiple receive beams, an angular estimate could be obtained from a single (mono) pulse, and used to update the direction of the antenna beam. The concept of monopulse dates back to the 1940s and saw much progress during the 1950s [20], [21]. Monopulse is still in much use today, and its supremacy in angular tracking has only recently been challenged by the introduction of digital arrays in combination with techniques such as ESPRIT [22]. We will devote more attention to monopulse in Chapter 7.

In Chapter 4 we discuss non-coherent processing of pulses in a special case of pulse compression. Developed in the mid 1950s [23], pulse compression is a technique where a longer pulse is processed to produce a synthetically short pulse, yielding both high range resolution and high transmit energy. Prior to this, the

range resolution was limited by the duration of the pulse. Pulse compression does however often require coherent modulated pulses, something the magnetron will not provide. This is in part the motivation for our study in Chapter 4.

Coherent and modulated signals are typically generated at lower power levels, and then amplified. The cavity magnetron can be regarded as a high power oscillator and is not an amplifier in that sense. It therefor became more and more replaced by amplifying tubes such as the broadband traveling-wave tube (TWT), and the klystron, much due to [24], published in 1953. However, both the klystron and the TWT date back to before WWII [25], [26]. In recent years solid state amplifiers have become more common in radar transmitters [27].

Coherent transmissions also came to allow Doppler processing between pulses, known as pulse Doppler radar. In pulse Doppler radar, a phase change[1] between pulses provides a means of estimating the radial velocity between the radar and target. This is not only useful when tracking an object, but also helps in detecting targets since the target velocity is often an important discriminating feature between targets of interest and background clutter. The first operational pulse Doppler radar was the AN/DPN-53 target seeker, developed in the late 1950s [28].

As the secrecy associated with radar began to fade after WWII, many important works relating to the theory of radar became published. Two such highly influential reports were released in the same issue of IRE Transactions on Information Theory in 1960, and are of particular relevance to this thesis. The first, is a 1947 Rand Corporation report by Jess Marcum [29]. In [29], the mathematical relations between detection probability, false alarm probability, the number of integrated pulses and the SNR is outlined. The random nature of the signal strength was with regards to noise and clutter, and the target was assumed to have fix radar cross section (RCS). In the second report (also Rand Corporation), written by Peter Swerling in 1954, this was extended to targets with fluctuating RCS [30]. Four models were proposed which have become known as Swerling targets I, II, III and IV. These are discussed in Section 2.7. In Chapter 5, we look at the detection probability when integrating pulses for a special case of moving Swerling targets.

When the received echo is in some way "modulated" by the target, such as the fluctuations described by Swerling, different types of targets can become distinguishable from each other. At least to some extent. One of the first experiments, if not the first experiment, to demonstrate this phenomenon was performed in 1923 by a Mr Brard, a French amateur radio operator. In his experiment, a vertical antenna was suspended in a tree and connected to the ground via an alarm clock. He illuminated this with a radar device of his own design, and could in an earphone hear the ticking of the clock [31]. This was only heard

[1] A phase change over time in sinusoidal signals is equivalent to a frequency shift. This shift in frequency arises when there is motion between transmitter, target and the receiver and is known as the Doppler effect.

when he directed the antenna beam towards the clock. He thereby concluded that the electromagnetic waves had been reflected, proving the concept of what would later be known as radar. At the same time he had, perhaps unknowingly, also demonstrated that attributes of the target could be picked up by the wave to the extent that the target could be recognized at the receiver. Unfortunately his experiments did not impress higher authorities [31].

As the use of radar grew during and after WWII, so did the need for automatically recognising targets, known as automatic target recognition (ATR) [32]. ATR has recently seen many contributions from the field of machine learning, and deep learning in particular. In our study in Chapter 6, we look at a special case of ATR for drones.

Finally, and perhaps a bit late in this historical narrative, we draw our attention to the word *radar*. As we mentioned in the beginning of this section, radar was developed simultaneously in several countries. Naturally they all had their own terminologies. Some examples are [33]:

- radio direction finding (RDF), Great Britain
- funkmessgerät (FuMG), Germany, “wireless gauge”
- radio detector telemetro (RDT), Italy, where telemetro can be translated as “distance gauge”
- радиоуловитель самолётов, or radio ulovitiel samaliotov (RUS), Russia, “radio aircraft catcher”
- détection électromagnétique (DEM), France, “electromagnetic detection”
- 電波探信儀 or denpa tanshingi, Japan, “radio wave probe” [34]
- ekoradio (ER), Sweden, “echo radio” [11]

The American term *radar*, stands for radio detection and ranging (RADAR) [35, Ch. 4]. It was coined in 1940 by the United States Naval Research Laboratory, and would soon become the standard international nomenclature. The fact that it is a palindrome makes it extra fitting considering radar’s association to reflections. Despite it being an acronym it is often used as a noun, and we will in this thesis continue with the noun notation “radar” rather than the acronym notation “RADAR”.

2.1.2 Applications

Had I known what you were going to do with it I would never have invented it![1]
— Sir Robert Alexander Watson-Watt, one of many inventors of radar.

The capability to detect and track targets from far distances, in different weather conditions and without the need of daylight, has made radar nearly indispensable in modern warfare. Radar's history as a military technology is no secret. It can even be inferred from early product names such as the Japanese night fighter radar "Gyoku-3", originating from gyokusai, a term for "honorable suicide" [13], or the Russian abbreviation RUS for "radio aircraft catcher" [12]. Some application areas in military radar are:

- **Search and Surveillance radar**. Many systems fit into this category. These radars are designed to search larger volumes in space and may be used for both surveillance and navigation. **Early warning radar** is a special type of surveillance radar designed to give early warning of hostile activities. They are characterised by larger antennas and long range capability. The aforementioned Chain Home along the British eastern and southern coasts was an example of such a system [15]. These can also be airborne.

- **Missile guidance and fire control radar**. Designed to be used as external "aiming" for missiles and guns. It provides estimates of target coordinates, e.g. in range, azimuth, elevation and radial velocity. An example is the CelsiusTech radar and optronic sensors (CEROS) [36].

- **Target seeker radar**. These radars are mounted in the front of missiles and designed to search for/acquire, and track single targets until impact. These radars need to be relatively small in size and therefore operate at higher frequency bands. An example is the aforementioned AN/DPN-53 target seeker used in the Boeing CIM-10 Bomarc, the first long range surface to air missile [28].

- **Imaging radar**. Used primarily for surveillance, these often airborne radars are designed to produce near image quality resolution of the illuminated area. A prominent technique for this is synthetic aperture radar (SAR) [37]. In SAR, radar echoes are sampled while an aircraft or satellite moves along a set course. The data collected over this course can then be processed as if it was measured with a wide synthetic aperture, yielding very high angular resolution.

[1] Allegedly said when he was pulled over for speeding by Canadian police in 1956. The policeman had used a radar gun.

Despite radars strong military history, some of the very first applications were for civilian use, such as Hülsmeyer's radar, used in collision avoidance for ships [10]. Today, the list of radar applications within the civilian sector is long and perhaps more diverse than that of the military sector. We here only present a limited selection of some of these areas.

- **Air traffic control radar**. This includes systems for detecting aircraft and vehicles on the ground, guiding landing aircraft, monitoring in the airspace around airports as well as traffic in larger search volumes. The different systems operate in different frequency bands, ranging from the L-band for longer ranges to the K_u band for shorter ranges.

- **Weather radar**. Used in both weather forecasting and in aircraft for detecting upcoming turbulence. At certain wavelengths, reflections from rain, snow and hail produce "maps" of the precipitation. The choice of frequency depends on application and weather radars operate mostly in the S-band, C-band and X-band.

- **Space based radar**. This includes imaging of planetary surfaces, e.g. satellite based SAR in the L-band and S-band.

- **Automotive radar**. Various applications such as collision warning sensors. Designated bands for automotive radar are found in the K-band and W-band.

- **Traffic radar**. May be used for traffic management where visual light cameras will fail or when the target velocity is to be estimated, such as radar speed guns.

- **Medical radar**. Used for monitoring patients, such as fall detection, and medical imaging. Compared to X-rays, 10^{16} to 10^{20} Hz, medical radar offers poorer spatial resolution but are harmless in comparison. The temporal resolution in medical radar is however high, and it can therefore be used to monitor fast events and over longer periods of time.

As for the studies presented in this thesis, the intended applications are primarily for military systems. However, Chapter 4 concerns non-coherent pulse compression, and is applicable to any situation where non-coherent illumination yields different intensity levels, at different frequencies, for different types of targets. The study in Chapter 5 concerns detection of weak targets. It is applicable to any system that can produce "A-scope" radar samples, i.e. practically all conventional radar systems, including bistatic and passive radars, see Section 2.2. In Chapter 6 we look at methods for distinguishing drones from non-drone targets. This is applicable to surveillance radar, and is as relevant to military surveillance

as it is to civilian surveillance. Our study in Chapter 7 is specific and relates to fire control radars for tracking in multipath interference.

2.1.3 Radar Frequency Bands

The limitations of a radar system is to a large extent set by the size of the antenna, the transmit power and frequency. Higher frequencies allow wider bandwidths and therefore higher resolution in range, see Section 2.2. Higher frequencies also yield larger antenna apertures relative to wavelength, and therefore higher resolution in angle, see Section 2.5. However, high frequencies also imply faster oscillations, which depending on the medium in which the wave propagates, may require more energy. This results in increased attenuation and consequently reduced range. The level of the medium absorption (attenuation) and reflection (e.g. from rain drops) depends on both weather and frequency. Although inferior in spatial resolution, lower frequency radar, e.g. VHF and UHF radar, is less sensitive to weather effects and may also be used for detecting stealth targets, as "stealth" is in this context often conditioned on the use of higher frequencies [38].

Much of the overall characteristics of a radar system can thus be implied from the frequency at which it operates. During WWII, letter designations were given to different frequency bands as a means to efficiently communicate these characteristics. The designations were set gradually as the technology developed, resulting in a nomenclature that may appear confusing, e.g. the HF, VHF and UHF bands are the three lowest frequency bands. The phrase "King Xerxes Seduced Lovely Princess" arose as a means of remembering the order of the K,X,S,L and P bands. In today's nomenclature, the P band (216 - 450 MHz) is incorporated into the VHF band. The disposition of the electromagnetic spectrum for radar frequencies, according to the Institute of Electrical and Electronics Engineers (IEEE) standard nomenclature, is given in [39] and shown in Table 2.1.

Table 2.1: IEEE Radar Frequency Bands [39]

Name	**Frequency**	**Wavelength**
HF	3 - 30 MHz	100 - 10 m
VHF	30 - 300 MHz	10 - 1 m
UHF	300 - 1000 MHz	1 - 0.3 m
L	1 - 2 GHz	30 - 15 cm
S	2 - 4 GHz	15 - 7.5 cm
C	4 - 8 GHz	7.5 - 3.7 cm
X	8 - 12 GHz	3.7 - 2.5 cm
K_u	12 - 18 GHz	2.5 - 1.7 cm
K	18 - 27 GHz	1.7 - 1.1 cm
K_a	27 - 40 GHz	1.1 - 0.75 cm
V	40 - 75 GHz	7.5 - 4 mm
W	75 - 110 GHz	4 - 2.7 mm
mm*	110 - 300 GHz	2.7 - 1 mm
THz	300 - 1000 GHz	1 - 0.3 mm

* Not to be confused with millimeter-wave radar operating at 30–300 GHz, wavelength 1-10 mm.

2.2 Basic Principles

In radar, a range r_0 is calculated from a time delay t_0 between a transmitted signal and a received echo as

$$r_0 = \frac{ct_0}{2}, \tag{2.1}$$

where c is the speed of light. The method for obtaining t_0 varies with different types of radars. Pulsed radars transmit a pulse, after which point the receiver begins to register the received power. Echoes will then appear as an increase in received signal power. The time delay is calculated from the time of transmission to the peak of these echoes. Two targets that are within a distance of

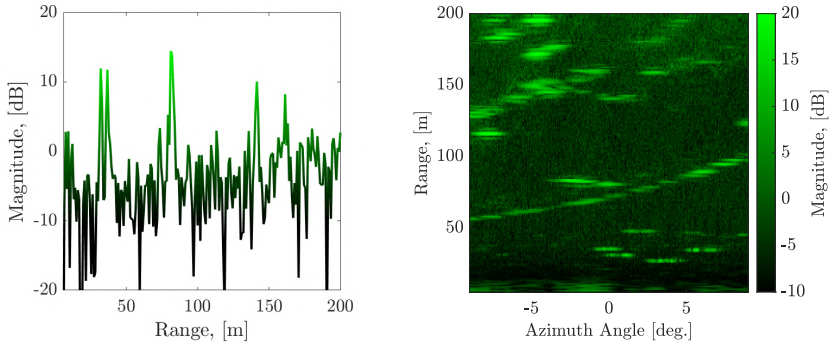
$$\partial r = \frac{c\tau}{2} \tag{2.2}$$

where τ is the pulse duration or pulse width, will appear as a single echo and cannot be resolved unless a technique known as pulse compression has been used, see Section 2.3. In this case the resolution is instead determined by the compressed pulse width $\tau' = 1/B$ where B is the pulse bandwidth. At a certain point the transmitter will transmit another pulse and the process is repeated. The time between two pulses is the pulse repetition interval (PRI), the inverse of which is the pulse repetition frequency (PRF). The PRI sets the maximum time delay, or maximum range, that can be unambiguously detected as $R^{\text{max}} = c/(2\text{PRF})$.

An alternative is the continous wave (CW) radar which transmits and receives continuously. There are thus no pauses for a receiver to detect echos. Instead the time delay is obtained by correlating or demodulating the received signal with the transmitted signal. The obtainable resolution in time delay, or rather the obtainable resolution in range ∂r, is then determined by the bandwidth B of the transmitted signal as

$$\partial r = \frac{c}{2B}. \tag{2.3}$$

Without some form of modulation the range resolution is therefore nonexistent, i.e. a target can be detected but the distance to it cannot be inferred. Many of the early radar systems were of this type, where objects were detected by noting a Doppler shift in the CW signal. A common CW radar waveform for obtaining spatial resolution is the frequency modulated continuous wave (FMCW). This is presented in Section 2.9. We note that (2.3) coincides with (2.2) if $B = 1/\tau$. This is true for rectangular shaped pulses if we define B as the 4 dB bandwidth. As mentioned, the resolution in (2.3) is also true for pulse compression systems where



(a) A-scope/display (b) B-scope/display

Figure 2.1: Examples of an A and B scope/display. These measurements were obtained with the FMCW radar presented in Section 2.9. The plot in (a) corresponds to ca 0 degrees azimuth angle in (b).

B is the modulation bandwidth of the transmitted pulse.

CW radar has the advantage of distributing energy over time yielding lower peak transmit power than what a pulsed system would require in order to transmit the same amount of energy. It is the transmitted energy that ultimately limits the detectable range of a given target. Monostatic CW radar will however suffer from interference from the transmitted wave in the receiver. The level of isolation that is achievable between the transmitter and receiver puts a practical limit to how much energy a monostatic CW system can transmit. A compromise between these two approaches is pulse compression, see Section 2.3.

Another type of radar is passive radar, or passive coherent location (PCL) [40], and operates on CW principles. These are bistatic systems that utilise transmitters of opportunity, such as other radar stations, television and radio broadcasting stations, telecommunication systems etc. Unlike regular bistatic radar, the transmitter operates independently of the receiver and no cooperative synchronisation is necessary. Instead an additional reference channel/antenna is directed at the transmitting source. The signal from this is then correlated with the received signal yielding a time delay from which a range can be inferred.

Whether a radar system is active, passive, bistatic, monostatic, pulsed or of CW type, the received signal can be processed to produce an A-scope ar A-display view [1]. An example of this using an FMCW radar is shown in Fig. 2.1a. A B display shows the received signal power as a function of both range and angle [1]. An example of this is shown in Fig. 2.1b obtained with the same FMCW radar. Echoes from targets are seen in Fig. 2.1 as relatively strong peaks. The signal strength of

such peaks are discussed in Section 2.2.1 and the basic steps in processing radar data is discussed in Section 2.2.2.

2.2.1 The Radar Equation

The radar equation can be used to determine the signal strength of a given target. A perhaps more interesting quantity is the signal strength relative to the background noise, since it is this ratio that will determine how well a target can be distinguished from noise. The received SNR in monostatic radar can be expressed as

$$\text{SNR} = \frac{P_r}{P_n} = \frac{P_t G_t G_r G_p \lambda^2 \sigma}{(4\pi)^3 F_n P_n' L L_a R^4} \tag{2.4}$$

where

- P_r is the received signal power and P_t is the transmitted power.
- P_n is the received noise power.
- G_t and G_r is the transmit and receive antenna gain respectively, and will depend on angle and frequency.
- G_p is a processing gain from matched Filtering, see Section 2.3. This is sometimes omitted from the radar equation since it is a synthetic gain obtained in the receiver.
- λ is the wavelength.
- σ is the RCS and may depend on many factors. It may even be considered random, see Section 2.7.
- F_n is a noise factor and accounts for the difference between the theoretical noise power P_n' and the observed noise power.
- L is a general loss that accounts for various factors such as the transmission and receive losses, processing losses etc.
- L_a is the medium absorption or attenuation loss, and will depend on frequency and weather (and therefore also the range).
- R is the range.

For a derivation of (2.4) the reader is referred to [3, Sec. 1.2]. The receiver noise P_n typically consists of thermal noise and is often modeled as that from a resistor

with temperature T_s. This will give a theoretical noise power [3, Sec. 1.2]

$$P_n' = k_{\mathrm{B}} T_s B_n \tag{2.5}$$

where $k_{\mathrm{B}} = 1.38 \times 10^{-23}$ W/(HzK) is Boltzmann's constant and B_n is the noise bandwidth defined as

$$B_n = \frac{1}{|H(f_c)|^2} \int_{-\infty}^{\infty} |H(f)|^2 df \tag{2.6}$$

where $H(f)$ is the frequency response of the receiver filter and f_c is the center frequency of $H(f)$. The value of T_s takes several temperatures into account such as that of the receiver, the apparent temperature of the sky etc. [3, Sec. 1.2].

The SNR in (2.4) is a rather general expression and can be simplified depending on what aspects are under consideration. In this thesis we will use a model of the received power of the form

$$P_r = |C|^2 \frac{\lambda^2 \sigma}{R^4} \tag{2.7}$$

where C is some arbitrarily chosen value. The modeling of σ is discussed in Section 2.7. We model the noise as zero mean additive white Gaussian noise (AWGN) where the power is given by the variance σ_w^2. This also accounts for other sources of noise such as digital quantisation errors and so forth. This gives us the SNR as

$$\mathrm{SNR} = \frac{P_r}{P_n} = \frac{|C|^2 \lambda^2 \sigma}{\sigma_w^2 R^4}. \tag{2.8}$$

Depending on scenario, the power from background clutter and interference may have a much stronger power level than the receiver noise. It will therefore be the signal-to-clutter-plus-noise ratio (SCNR) or the signal-to-interference-plus-noise ratio (SINR) that ultimately determines how well a target can be detected. Both interference and clutter come from real signals but are sometimes modeled as random variables making them not much different from noise in that sense. Clutter refers to unwanted reflections from ground, sea, precipitation, etc. [1], whereas interference typically refers to signal power from other targets or other emitting sources such as jammers. We will in this thesis for the sake of simplicity mostly use of the SNR with the understanding that this could be replaced with the SINR or the SCNR under different modeling assumptions.

2.2.2 The Signal Processing Chain

The advances made in radar since WWII can be attributed to many factors. As for the signal processing, the development from analogue to digital processing is due much credit. Beginning in the late 1960s it has continued to evolve to this day with higher sampling frequencies, higher processing power and analog to digital converters (ADCs) moving closer and closer to the antenna, trends not unique to radar. In this section we will outline some of the key processing steps in a radar receiver. A typical radar system may involve the following:

1. Transmission of signal, including waveform generation, upconversion to radio frequency and amplification.
2. Reception and analogue downconversion to intermediate frequency (IF).
3. Conversion to digital domain with ADCs.
4. Signal processing including matched filtering.
5. Detection and clustering.
6. ATR and classification.
7. Association with previous detections/clusters and tracking.

Note that 1 is in the transmitter and 2-7 are in the receiver. These steps are usually executed in the given order although this may vary to some extent. As an example, in Chapter 5 we track before detection. These steps will also vary depending on the application and type radar.

This thesis concerns steps 4-7 and we therefore begin with the output from the ADC in step 3. The data sampled during a processing interval can be represented as a three dimensional tensor, sometimes referred to as the "data cube". For coherent radar, it is common to refer to this interval as a coherent processing interval (CPI). One dimension is the channels where each channel is the output from an ADC. Samples along this dimension are obtained instantaneously but may be spatially diverse. Another dimension is the "fast time" dimension. This contains data sampled during a PRI. The last is the "slow time" dimension and contains samples from different PRIs. In CW and PCL radar, a similar structure can be formed where a longer sequence is segmented into a structure analogue to the data cube [40]. The data cube is illustrated in Fig. 2.2.

The purpose of the signal processing in step 4 is to enable detection of targets in range, velocity and angle, or simply to produce images in these dimensions. Range information is obtained from processing along the fast time dimension. This is discussed in Section 2.3. Velocity information is obtained from processing along

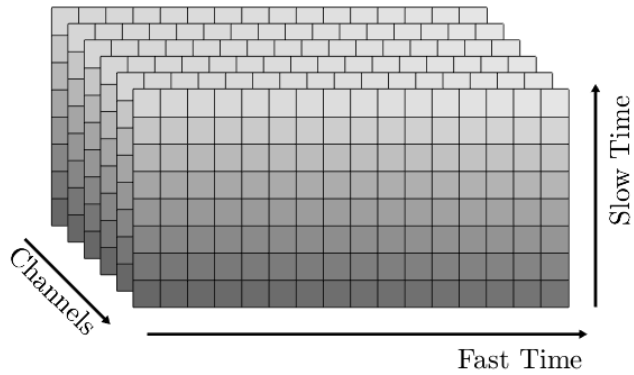


Figure 2.2: Example of the data cube for a system with six ADCs and eight pulses in a CPI. Range information is obtained from processing along the fast time dimension. Velocity information is obtained from processing along the slow time dimension and angular information is obtained from processing along the slow time and/or channel dimension.

the slow time dimension. This is discussed in Section 2.4. Angular information is obtained from processing along the channel dimension and/or along the slow time dimension if the antenna beam has moved during the CPI. This is discussed in Section 2.5. An example is the B-scope image in Fig. 2.1b. In this case there was only one channel and the range and angle matrix in Fig. 2.1b is the result from processing a single fast time/slow time data matrix.

The detection process in step 5 is discussed in Section 2.6. The terms *detection* and *classification* can be a bit ambiguous and we therefore define these as follows. By *detection* we refer to the decision of whether a target is present or not - regardless of its type. By *classification* we refer to labeling a target e.g., as either *drone* or *non-drone*. We use the term *identification* when distinguishing between different targets of the same class. The classification and identification steps are both referred to as *target recognition* or ATR. A target will often span more than one cell in range, angle and/or velocity, yielding multiple detections per target. These may be clustered to produce a single detection.

The ATR in step 6 is not always necessary or even possible. Classifying different types of targets is the topic of our discussions in Chapter 6. The association of new detections with old detections in step 7 can take different forms and is often aided by the tracking procedure. A common tracking method is the Kalman filter which we use and present in Chapter 7.

2.3 Matched Filtering

A received echo from a point scatterer[1] will approximately have the same pattern as the transmitted pulse, only time delayed, time reversed and scaled in magnitude. Knowing this, we can construct a receiver filter that gives a high valued output for signal segments that have this pattern, and relatively low outputs otherwise. Filters that maximize this SNR are known as matched filters [41]. The filter is "matched" since it will have the same pattern/shape as the transmitted waveform. The reeived filtered signal is

$$\tilde{\boldsymbol{x}}[n] = \sum_{k=-\infty}^{\infty} \boldsymbol{h}^*[n-k]\boldsymbol{x}[k], \tag{2.9}$$

where $\boldsymbol{h}[n]$ is the matched filter, * denotes complex conjugate and $\boldsymbol{x}[k]$ is the sampled signal. A special case of matched filtering is pulse compression [23]. In pulse compression, the pulse waveform is designed such that the autocorrelation function

$$\boldsymbol{R}_{\boldsymbol{hh}}[n] = \sum_{k=-\infty}^{\infty} \boldsymbol{h}^*[k-n]\boldsymbol{h}[k] \tag{2.10}$$

approximates the Kronecker delta function, i.e.

$$\frac{\boldsymbol{R}_{\boldsymbol{hh}}[n]}{\boldsymbol{R}_{\boldsymbol{hh}}[0]} \approx \delta[n] = \begin{cases} 1, & \text{if } n = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{2.11}$$

By so doing we concentrate, or compress, the signal energy to a shorter interval, but only for signal segments that are matched to the transmitted waveform. This not only increases the peak SNR, but it also improves the range resolution. Note that the length of $\boldsymbol{h}[n]$ is the same as the pulse length τ, whereas the "length", or synthetic pulse length of (2.11) is ideally one sample. The gain in range resolution and SNR is characterised by the time-bandwidth product, or the pulse compression ratio, $B\tau$ [3, Sec. 5.2].

The resulting radar system can then be thought of as the equivalent of having a system with a shorter pulse length but with the same amount of transmitted energy, i.e. higher peak power, thus improving the maximum range. As indicated in (2.11) this will only be approximately true. For the approximation in (2.11) to become an equality, an infinite bandwidth would be required. The limited bandwidth of $\boldsymbol{h}[n]$ will thus limit the achievable range resolution according to (2.3). The energy

[1] A target with a spatial span that is much smaller than the pulse length.

leakage in (2.10) for all $n \neq 0$, known as sidelobes, will in practice have a span of $\pm\tau$, see the plot for $\tilde{\boldsymbol{x}}$ in Fig. 2.3. A relatively weak target can become covered by the sidelobes of a neighboring strong target and the detection thus becomes conditioned on the relative signal strengths of all targets within $\pm\tau$.

There are many methods for obtaining waveforms with the desired pulse compression properties. In coherent radar, it can be achieved with intra-pulse modulation, where the phase changes with time. One such modulation is linear frequency modulated pulse compression [42]. Used in chirp radar, the frequency is linearly changed over a bandwidth B during the pulse length τ. This is illustrated in Fig. 2.3a where a synthetic pulse length τ' is achieved. In non-coherent radar where such modulations cannot be formed, pulse compression can be achieved with on-off keying [43]. This is illustrated in Fig. 2.3b. By transmitting a sequence of pulses which are either on or off, a longer "pulse" is formed. This sequence is chosen such that the autocorrelation has the desired properties. In Fig. 2.3b we have used an m-sequence [44]. We discuss this in more detail in Chapter 4 where we also present a new form of non-coherent pulse compression using ANNs.

2.4 Doppler Processing

The Doppler effect, described in 1842 by Christian Doppler, arises when there is motion between a source/transmitter and an observer/receiver. An observed signal will then appear scaled in time by a factor β, when compared to the emitted signal. For monostatic radar, this factor is

$$\beta = \frac{c + v_r}{c - v_r} = 1 + \frac{2v_r}{c - v_r} \approx 1 + \frac{2v_r}{c} \tag{2.12}$$

where v_r is the relative radial velocity between the radar and target, measured as positive if these are moving towards each other. For a sinusoidal wave, this scaling in time is equivalent to a frequency shift since

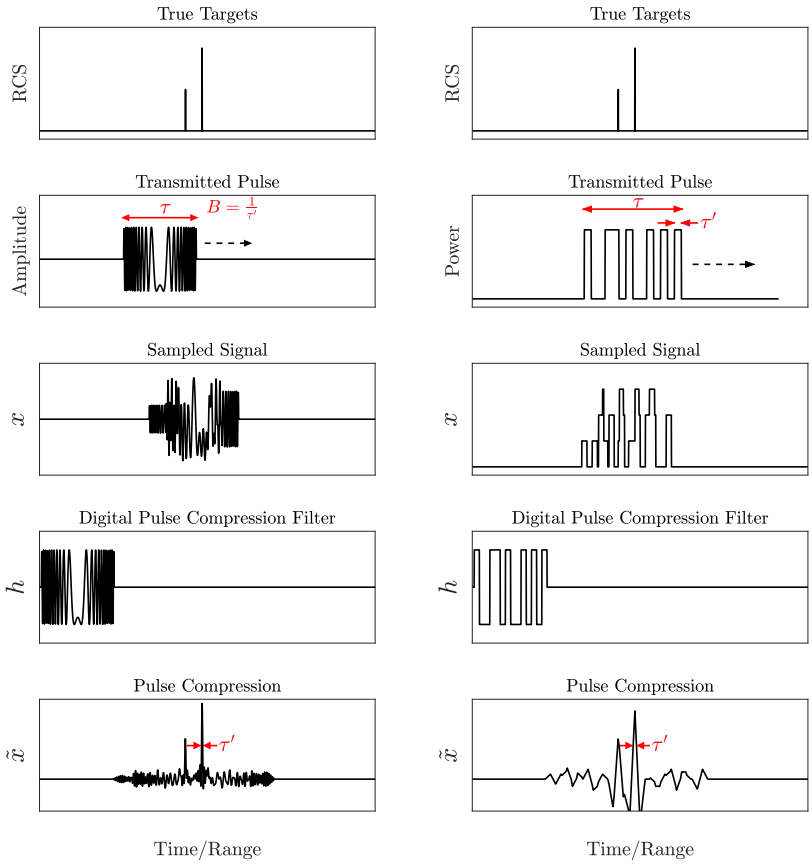
$$\sin(2\pi f_c \beta t) \approx \sin\left(2\pi f_c \left[1 + \frac{2v_r}{c}\right] t\right) = \sin\left(2\pi \left[f_c + f_c \frac{2v_r}{c}\right] t\right). \tag{2.13}$$

The apparent shift in frequency

$$f_{\mathrm{D}} = f_c \frac{2v_r}{c} = \frac{2v_r}{\lambda_c} \tag{2.14}$$

is the Doppler frequency, and we note that it contains information about the relative velocity of the target. We will refer to this as the fast time Doppler frequency.

In coherent processing, the Doppler frequency also appears as a phase shift



(a) Linear chirp

(b) on-off keying

Figure 2.3: Examples of coherent (a) and non-coherent (b) pulse compression. The horizontal line in all plots denote the value zero. In (a) the pulse is linearly modulated in frequency over a bandwidth B were $\tau B = 90$. Note that the figures illustrate the baseband waveform. In (b) a series of short pulses are transmitted with on-off keying in 14 slots. The bandwidth is set by the duration of each sub-pulse and is $B = 1/\tau'$. The compression ratio is thus $\tau B = \tau/\tau' = 14$. The notation $\boldsymbol{x}$, $\boldsymbol{h}$ and $\tilde{\boldsymbol{x}}$ are in accordance with (2.9).

between pulses. We refer to this as the slow time Doppler frequency. The slow time Doppler frequency is exploited in pulse Doppler radar. Assume that the sampled value at pulse 0 at time t_0 is complex valued, denoted $s(0, t_0)$. Further assume that the time t_0 corresponds to a range of a target with radial velocity v_r. Due to this velocity, the target will have moved a radial distance

$$\partial r = v_r T_p \tag{2.15}$$

at the next pulse at time t_0, where T_p is PRI value. Here our notation is such that we reset the time at each pulse, i.e. we are looking at one column in Fig. 2.2. The next pulse will thus be shifted in time by $2\partial r/c$, or in phase by

$$s(1, t_0) = s(0, t_0)e^{j2\pi f_c \frac{2\partial r}{c}} = s(0, t_0)e^{j2\pi f_c \frac{2v_r}{c} T_p}, \tag{2.16}$$

where f_c is the carrier frequency. Likewise for pulse $N-1$ we get

$$s(N-1, t_0) = s(0, t_0)e^{j2\pi f_c \frac{2v_r}{c} T_p (N-1)} \tag{2.17}$$

This assumes that the pulses are coherent, i.e. no random or unknown change in phase between pulses other than that caused by v_r. It also assumes that the total change in range, $(N-1)\partial r$, is negligible compared to the range resolution of the system. Examining (2.16) and (2.17), we note that v_r gives rise to a frequency over pulses at a given time t_0. We denote this slow time Doppler frequency as

$$\nu_{\mathrm{D}} = f_c \frac{2v_r}{c} T_p = \frac{2v_r T_p}{\lambda_c}, \tag{2.18}$$

which is the same frequency as the fast time Doppler in (2.14), if we multiply ν_{D} with the "sampling frequency" $1/T_p$.

Using the discrete time Fourier transform over the set $s(0, t_0), ..., s(N-1, t_0)$, we obtain the sinc-like function

$$\begin{aligned} S(\nu) &= \left| \sum_{n=0}^{N-1} s(n, t_0) e^{-j2\pi\nu n} \right|^2 \\ &= \left| \sum_{n=0}^{N-1} s(0, t_0) e^{-j2\pi(\nu-\nu_{\mathrm{D}})n} \right|^2 \\ &= |s(0, t_0)|^2 \left| \frac{\sin(\pi[\nu - \nu_{\mathrm{D}}]N)}{\sin(\pi[\nu - \nu_{\mathrm{D}})} \right|^2 \end{aligned} \tag{2.19}$$

for $-1/2 < \nu \leq 1/2$ where $\nu = fT_p$. The achievable frequency resolution in

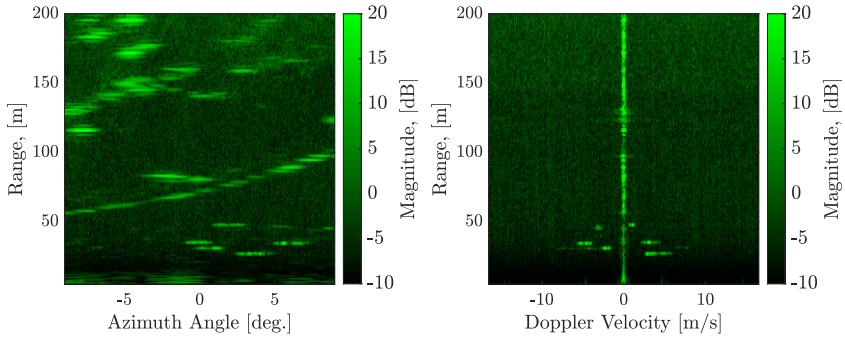
normalized frequency is then $\partial\nu \approx 1/N$. Using (2.18), we obtain the highest achievable radial velocity resolution as

$$\partial\nu_D \approx \frac{1}{N} \rightarrow \partial v_r \approx \frac{\lambda_c}{2NT_p}. \tag{2.20}$$

We cannot distinguish between integer shifts of a wavelength between pulses, and we therefore have velocity ambiguities when ν_{D} reaches the Nyquist frequency at $\pm 1/2$. This gives us the unambiguous velocity range

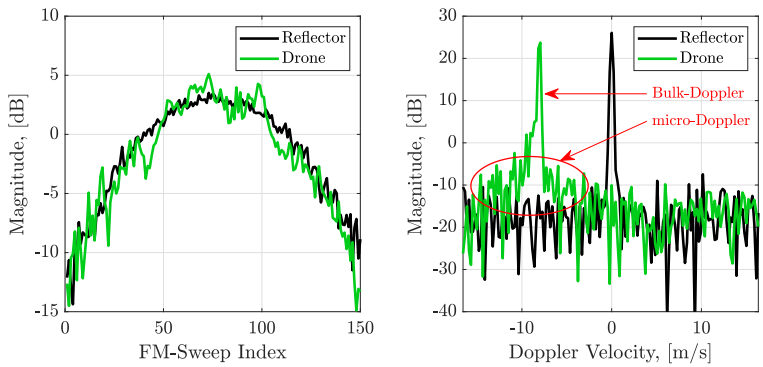
$$-\frac{\lambda_c}{4T_p} < v_r \leq \frac{\lambda_c}{4T_p}. \tag{2.21}$$

The Doppler spectrum may thus be obtained with a discrete Fourier transform (DFT) over pulses (slow time). An example of this is shown in Fig. 2.4. The targets with motion in Fig. 2.4 are a flock of birds. A target with many moving parts will have many velocity components with different values of v_r. This is known as micro-Doppler, and the term bulk-Doppler then refers to the velocity of the whole target. An example of such a spectrum is shown in Fig. 2.5. In Chapter 6 we use the micro-Doppler spectrum to distinguish between different types of moving targets.



(a) Before Doppler Processing (b) After Doppler Processing

Figure 2.4: Example of Doppler Processing. In (b) we have applied the DFT over azimuth (slow time dimension) in (a). Most targets are stationary and the energy from these are concentrated to the cells around 0 m/s in (b). A few targets at less than 50 m range are in motion and can easily be disguised. These are a flock of birds. These measurements were obtained with the FMCW radar presented in Section 2.9.



(a) Time Domain (b) Frequency Domain

Figure 2.5: FMCW radar measurement of a drone and a trihedral corner reflector using the sensor in Section 2.9. The time domain image in (a) shows the magnitude for 150 consecutive FM-sweeps (see Section 2.9). During this time the antenna is mechanically scanned. The reflector is stationary and appears as a peak at zero velocity in (b). The drone however is in motion, with bulk-Doppler velocity at -8 m/s. At the frequencies or velocities around this peak, we see micro-Doppler components above the noise floor at ca -20 dB. These come from reflections off the rotating propellers.

2.5 Antennas

A simple method to obtain the angular direction to the target, is to scan the antenna beam and select the angle at the peak signal value. In older systems this was often done by mechanically turning the antenna. In more recent years, methods of electronically steering the beam using phased arrays have become more dominant. Depending on radar type, it may be desirable to track a target once it is detected. Rather than continuously scanning around the target, we can find the peak of the main beam from a single (mono) pulse using a monopulse antenna. In Chapter 7 we exploit monopulse antenna technology and phased arrays. We will therefore give some background to these. We will use the following terminology [45].

- **Antenna directivity**. A hypothetical antenna that radiates equally in all directions is called an isotropic antenna. In practice however, an antenna will have stronger radiation in certain directions. The ratio between this radiation pattern and that of the hypothetical isotropic antenna, is called the antenna directivity. The directivity is typically given in decibels, or more specifically decibels relative to an isotropic antenna (dBi), and is a function of the spherical angles in azimuth and elevation. These angles are given with reference to the boresight direction, or the normal to the antenna surface. In this thesis we will mostly consider the directivity along a single angular dimension (azimuth or elevation). See example in Fig. 2.7.

- **Antenna gain**. The antenna gain G is related to the directivity D as $G = \eta D$, where $\eta \in [0, 1]$ is the antenna radiation efficiency.

- **Main beam**. The antenna gain will have a pattern with beams of different gains. The beam with the strongest gain is the main beam, and we will denote the direction of this beam as the reference direction or reference angle. The other beams are referred to as sidelobes.

- **Beamwidth**. Let the gain in the reference angle ϕ be $G(\phi)$. As we deviate from this direction the transmit gain, and reciprocally the receive gain, will diminish. The angle between the two angles where the gain is reduced by half is the half power beamwidth, or 3 dB beamwidth. In radar it is common to use the same antenna for both transmission and reception. The total gain is then $G(\phi)^2$. The 3 dB beamwidth for G^2 is the two way beamwidth and is smaller than the one way beamwidth of G.

A common type of antenna is the dipole antenna. The antenna elements in the array antennas discussed in Section 2.5.1 can be thought of as dipole elements. The radiation of a dipole antenna is briefly described in Fig. 2.6.

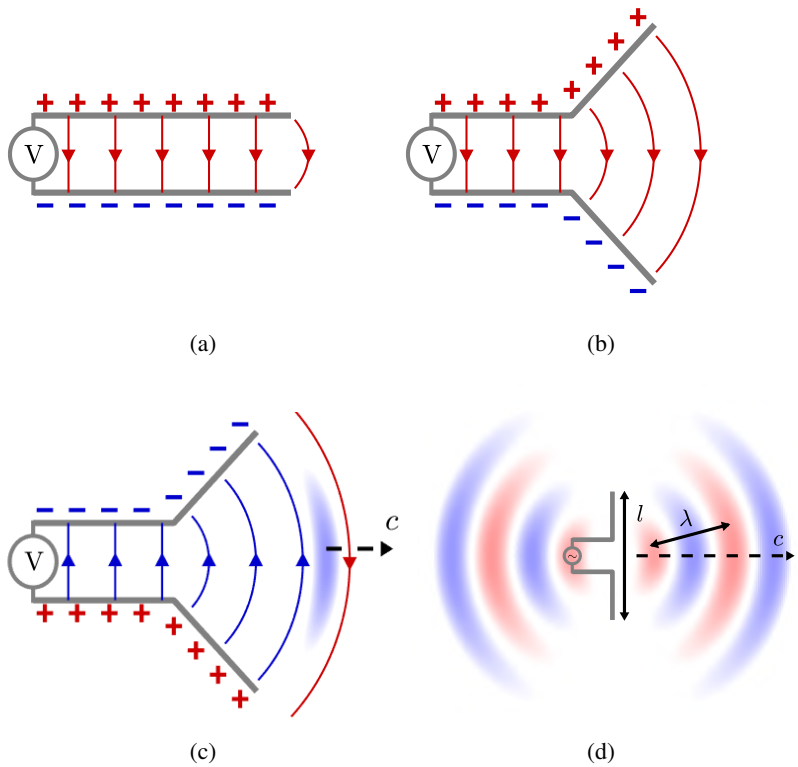


Figure 2.6: The dipole antenna. Note that the images are drawn out of proportion. (a) An electrostatic field occurs between two electrodes with opposite charge. At the edge, the field is "fringed" or bent. (b) By bending the electrodes, the fringed field becomes larger. (c) Switching the voltage across the electrodes will change the charge distribution in the electrodes, reversing the field. The change in the electric field will propagate from the electrodes at the speed of light, c, known as an electromagnetic impulse. (d) By frequently changing the voltage across the electrodes with frequency f, we generate an electromagnetic wave. Between two such "switches", the wave will propagate the distance of one wavelength λ, which is related to f and c as $c = f\lambda$. A dipole antenna is obtained by bending the electrodes to $\pm 90^{\circ}$, which spatially maximizes the emitted filed. The field strength will vary with aspect angle, the voltage level and f. Resonance is obtained when $l = n\lambda/2$ for integers $n \geq 1$. The field will then be the strongest perpendicular to the dipole. At this direction the directivity is 2.15 dBi for a half wave dipole where $l = \lambda/2$. From reciprocity, an electromagnetic wave that passes over a dipole antenna, will induce a voltage across the two electrodes in the receiving end.

2.5.1 Phased Array Antennas

Consider a one way scenario with a receiver and a single incoming signal. We will later look at the two way radar scenario in Chapter 7. Let the received signal originating from a distance r and angle θ at a single antenna element be

$$s_0(t) = \frac{A\sqrt{G'(\theta)}}{r^2} e^{j(\omega_c t - kr)} \tag{2.22}$$

where A is a signal amplitude, $\sqrt{G'(\theta)}$ is the amplitude gain of the antenna element, $\omega_c = 2\pi f_c$, $k = 2\pi/\lambda_c$ and f_c and λ_c is the frequency and wavelength respectively. Assume now that we have M antenna elements positioned according to the uniform linear array (ULA) geometry in Fig. 2.7.

Under the far field[1] approximation the angle to the target is the same for all elements and we can write the path difference between each element and the center of the array as $\partial r_m = ((M+1)/2 - m)d\sin(\theta)$. See Fig. 2.7. Further assume that we add a phase shift

$$\beta_m = e^{-jkmd\sin(\phi)} \tag{2.23}$$

to each antenna element. Ignoring noise, the sum of the received signal over all M elements is

$$\begin{aligned} s(t,\theta,\phi) &= \sum_{m=1}^{M} \beta_m s_m(t) = \sum_{m=1}^{M} \beta_m \frac{A\sqrt{G'(\theta)}}{r^2} e^{j(\omega_c t - k[r+\partial r_m])} \\ &= \frac{A\sqrt{G'(\theta)}}{r^2} e^{j(\omega_c t - kr)} e^{-jk(M+1)/2} \sum_{m=1}^{M} e^{jkmd[\sin(\theta)-\sin(\phi)]} \\ &\propto e^{j\omega_c t}\sqrt{G'(\theta)} \sum_{m=1}^{M} e^{jkmd[\sin(\theta)-\sin(\phi)]} \end{aligned} \tag{2.24}$$

where the final proportionality denotes a complex multiplication. The magnitude of (2.24) will be maximized whenever $kd[\sin(\theta) - \sin(\phi)] = n2\pi$ for integer n, at which point the sum in (2.24) is equal to M. Since $\sin(\theta) - \sin(\phi)$ has a span of at most 2, we can confine the periodicity of the maxima to 2π by setting $d = \lambda/2$

[1] Far field: When the range to the target is "sufficiantly" many wavelengths. What is sufficient depends on the scenario, see [46] for an electromagnetic analysis on the far field. In this geometric case, the approximation holds well for ranges beyond 100 λ.

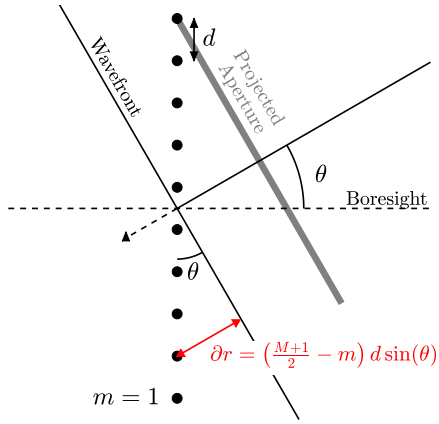


Figure 2.7: ULA structure with $M = 10$ elements marked as circles. These have index $m = 1, ..., M$ starting from the bottom. The distance between each element is d. The angle to the target is θ. Under the far field assumption, an incoming wave front can be regarded as a straight line. The projected aperture is $d(M-1)\cos(\theta)$.

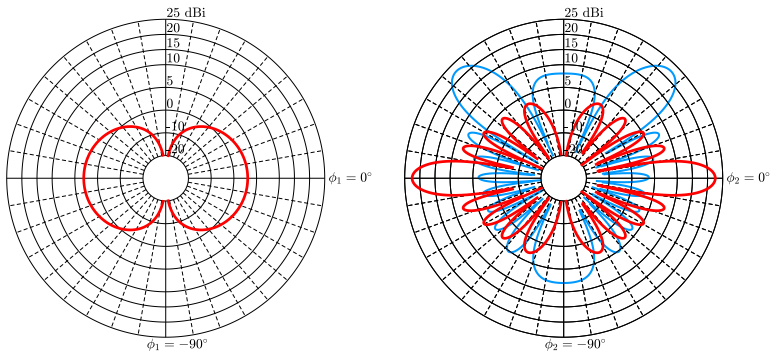
which gives us

$$kmd[\sin(\phi) - \sin(\theta)] = \left\{ d = \frac{\lambda}{2}, k = \frac{2\pi}{\lambda} \right\} = \pi m[\sin(\phi) - \sin(\theta)]. \quad (2.25)$$

The maxima occurs when $\phi = \theta$ and when $\phi = \pi - \theta$. We can eliminate this ambiguity by only considering a field of view of $\pm 90°$. This can in practice be achieved by limiting the directivity in $G'(\theta)$. The total gain is then

$$G(\phi) = G'(\phi) \left| \sum_{m=1}^{M} e^{jm\pi[\sin(\phi)-\sin(\theta)]} \right|^2 = G'(\phi) \left| A_f(\phi) \right|^2 \quad (2.26)$$

where $A_f(\phi)$ is the array factor of the phased array antenna. By setting the phase in β_m, we determine the reference direction and can thereby scan or steer the antenna beam. The vector $\boldsymbol{\beta} = [\beta_1, .., \beta_M]$ is therefore known as the steering vector. The aforementioned Eagle AN/APQ-7 radar is an example of a system that introduces these phase shifts from analogue delays in a waveguide [18]. A system that instead electronically sets this phase is known as an electronically scanned array (ESA) antenna. This eliminates practical problems with mechanically scanning systems



(a) Elevation, $\phi_2 = 0 \ \forall \ \phi_1$. Single Element

(b) Azimuth, $\phi_1 = 0 \ \forall \ \phi_2$. Ten element array

Figure 2.8: The far field antenna array directivity using dipole elements. In (a) the directivity of a half wave dipole is shown as a function of the elevation angle ϕ_1. Along the azimuth dimension ϕ_2, where $\phi_1 = 0 \ \forall \ \phi_2$, the dipole is theoretically omnidirectional with directivity 2.15 dBi. In (b) we see the directivity of a ULA as a function of azimuth, with half wavelength spacing between ten dipole elements. The phase difference, caused by the path difference to each element, will cause the total field from all elements to add up constructively or destructively depending on angle, creating nulls and peaks. In (b) we see two cases where the beam is steered to $0°$ (red) and $45°$ (blue). Note the slightly wider beamwidth in the latter case.

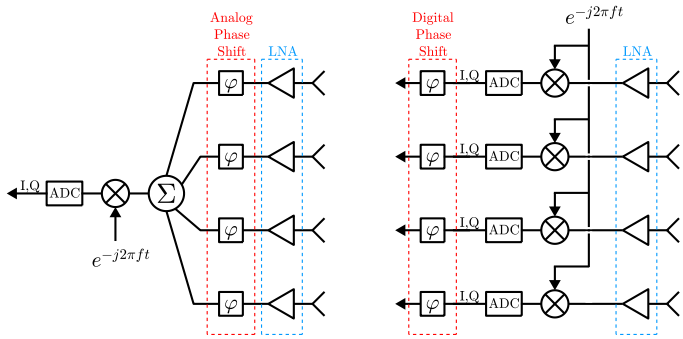
at a cost of an increased beamwidth. Let $G'(\phi) = 1 \ \forall \ \phi$ such that

$$G(\phi) = \left| \sum_{m=1}^{M} e^{jm\pi[\sin(\phi)-\sin(\theta)]} \right|^2 = \left| \frac{\sin\left(M\pi(\sin(\phi) - \sin(\theta))/2\right)}{\sin\left(\pi(\sin(\phi) - \sin(\theta))/2\right)} \right|^2. \quad (2.27)$$

Equating (2.27) to the half power level $M^2/2$ for $\theta = 0$ gives an approximate beamwidth

$$\phi_{3\text{dB}} \approx \frac{2}{M-1}, \quad (2.28)$$

i.e. it is inversely proportional to the array aperture $d(M-1)$. For other values of θ, the beamwidth will be inversely proportional to the projected aperture illustrated in Fig. 2.7. We can see this by taking a second order Taylor expansion of (2.27) at



(a) Analog ESA (b) Digital ESA

Figure 2.9: Example of an analog ESA (a), and a digital ESA (b) with four antenna elements. The figures illustrate the receiver path. Each antenna element has its own receive module in the form of a low-noise amplifier (LNA), making it an AESA receiver.

$\phi = \theta$ which gives us the approximation of the main beam as

$$G(\phi) \approx \left| M + \frac{a}{2}(\phi - \theta)^2 \right|^2 \tag{2.29}$$

where

$$a = M \cos^2(\theta) \left(\frac{\pi}{2}\right)^2 \left(1 - \frac{M^2}{3}\right). \tag{2.30}$$

Note that $G(\phi = \theta) = M^2$ and the first derivative at the peak of the main beam is zero. Equating (2.29) to the half power level $M^2/2$ gives us

$$\phi_{3\text{dB}} \approx 2 \frac{0.84}{\sqrt{M^2 - 3}} \frac{1}{\cos(\theta)} \propto \frac{1}{(M-1)\cos(\theta)} \tag{2.31}$$

for $M \geq 2$ and $\theta \in [-90°, 90°]$. The practical field of view for such phased array antennas is therefore often limited to $\pm 60°$, where the 3 dB beamwidth is increased by a factor 2 relative to boresight. Examples of the directivity of a phase steered ULA is shown in Fig. 2.8.

An ESA antenna, and can be either an active electronically scanned array (AESA) with amplifiers at each antenna element, or a passive electronically scanned array (PESA) with a single transmit module. Steering the receive reference angle can also be achieved digitally by sampling each antenna element, known as

a digital array, see Fig. 2.9. A digital array has two main advantages. Firstly, it enables simultaneous beamforming since we can evaluate the sum in (2.24) for arbitrary ϕ in the digital domain. Secondly, it also enables the use of the subspace direction finding techniques discussed in Chapter 7. Note that for radar, an arbitrary digital beam steering capability requires a uniform illumination of the entire field of view, since we will mainly receive reflections within the beamwidth of the transmit lobe. This will come at a cost of the transmit antenna gain since we cannot have both high antenna gain and wide beamwidth.

2.5.2 Monopulse

As mentioned in Section 2.5, monopulse antennas can be used to estimate the angle to the target from a single pulse. This assumes that the target is already within the beamwidth of the antenna, and that there is only one target within the same angular, range and velocity resolution interval. A notable exception to this occurs with multipath propagation which we discuss in Chapter 7. Monopulse antennas require at least two receiving channels in the one dimensional case (azimuth or elevation). These can be configured as either amplitude monopulse or phase monopulse. Amplitude monopulse is illustrated in Fig. 2.10a. Two receive beams are formed, e.g. by having two steering vectors in two separate channels in an ESA antenna. These are referred to as the sum channel Σ_x and the difference channel Δ_x, where the difference channel has a notch in the main beam forming two side beams with opposite phase. Due to this, a target that is off the reference direction will then have a positive or negative amplitude gain depending on the angle. The real part of the ratio

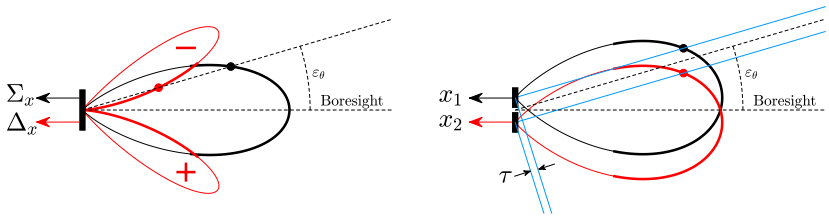
$$\eta^{\mathrm{A}} = \frac{\Sigma_x \Delta_x^*}{\Sigma_x \Sigma_x^*} \tag{2.32}$$

will be proportional to the offset between the reference angle and the true target angle ε_θ, cf. Fig. 2.10. By normalizing by the magnitude of the sum channel we remove the dependency of the signal strength[1]. Multiplying this ratio by a constant κ^{A} will together with the reference direction ϕ give the single pulse estimate of the target angle as

$$\hat{\theta}^{\mathrm{A}} = \phi + \kappa^{\mathrm{A}} \mathrm{Real}\left[\eta^{\mathrm{A}}\right]. \tag{2.33}$$

In a phase monopulse system with an ESA antenna, the receive antenna is divided into two subarrays, sampled in separate channels x_1 and x_2. This is illustrated in Fig. 2.10b. From a far field view, the gain to a target will be the same in both channels. The time delay, or phase in narrow band systems, will however be different due to the spatial offset between the two subarrays. We can compare

[1] This is only asymptotically true for large SNR values since the noise power in the denominator will reduce the ratio in (2.32)



(a) Amplitude Monopulse (b) Phase Monopulse

Figure 2.10: Different monopulse configurations. The thicker lines on the lobes indicate the beamwidth within which the monopulse angle error is approximately linear. In both cases the main beam is aligned with boresight. The difference channel Δ_x, is the sum of two overlapping beams with opposite phase.

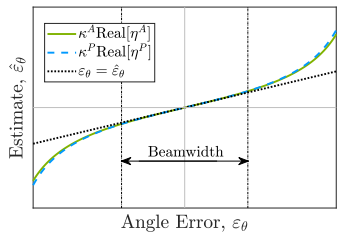


Figure 2.11: Angle error estimate for both amplitude (A) and phase (P) monopulse within the antenna beamwidth.

the phase by taking the ratio

$$\eta^{\mathrm{P}} = j\frac{x_1 - x_2}{x_1 + x_2}. \tag{2.34}$$

The real part of this will, much like η^{A} in amplitude monopulse, be proportional to the angular offset ε_θ in Fig. 2.10. The phase monopulse angle estimate is obtained as

$$\hat{\theta}^{\mathrm{P}} = \phi + \kappa^{\mathrm{P}} \mathrm{Real}\left[\eta^{\mathrm{P}}\right]. \tag{2.35}$$

We see in Fig. 2.11 that the estimated angle error has an almost linear relation to the true angle error within the beamwidth. We noted however in Section 2.5.1 that the beamwidth of the array factor in (2.26) depends on the reference angle ϕ. The factor κ^{A} in (2.33) and κ^{P} in (2.35) will therefore need to depend on ϕ to compensate for this.

2.6 Detection

Due to varying levels of clutter, noise and interference, a constant level threshold rule for the received signal power will in most cases be a poor detection criteria. Such a detector would produce many false alarms when the background signal level increases. In constant false alarm rate (CFAR) algorithms, it is instead the relative signal power that determines whether a sample should be classified as a detection or not. In CFAR processing, a sample or cell under test (CUT), is compared to a local average signal level obtained from neighboring reference cells/samples. To avoid having signal power from the CUT affecting the averaged reference value (assuming that a target is present at the CUT), guard cells are placed next to the CUT which are not used as reference. The reference cells are then obtained from a lagging and leading window outside the guard cells.

A typical one dimensional CFAR filter is shown in Fig. 2.12. The simplest form is the cell averaging CFAR (CA-CFAR), where the leading and lagging cells are averaged as

$$\bar{q} = \frac{1}{2L} \sum_{l=1}^{L} \boldsymbol{x}_{\text{lagg}}[l] + \boldsymbol{x}_{\text{lead}}[l] \tag{2.36}$$

where $\boldsymbol{x}_{\text{lagg}}[n] \in \mathbb{R}^{L\times 1}$ and $\boldsymbol{x}_{\text{lead}}[n] \in \mathbb{R}^{L\times 1}$ are the lagging and leading cells, respectively, and $2L$ is the window size. The CUT is declared as a detection if

$$x_{\text{CUT}} > T\bar{q}, \tag{2.37}$$

otherwise not, where x_{CUT} is the sample value of the CUT. The value of the threshold T sets the trade-off between false alarm probability and detection probability. For a stationary distribution of the noise, the false alarm rate will be constant for constant T. The detection probability will also depend on the target SNR. The false alarm probability, detection probability and SNR can be regarded as three degrees of freedom in the detection process, but where only two can be arbitrarily chosen when selecting T. The third is determined by the other two. This trade-off is given by the receiver operating characteristic (ROC) curve and is characterised by the overlap of the PDF of

$$x_{\text{CUT}}/\bar{q} \tag{2.38}$$

when the CUT contains a target, and the PDF of (2.38) when the CUT does not contain a target. Note that the normalization in (2.38) will typically make these distributions independent of the mean value of x_{CUT}. In practice the background statistics will not be stationary. Depending on the assumptions made

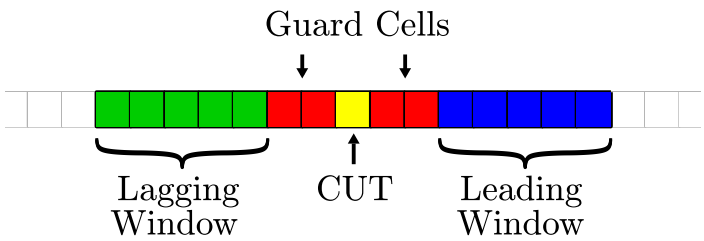


Figure 2.12: Example of a CFAR window.

on these statistics, the trade-offs between the false alarm probability and detection probability may be a rather complicated matter to determine.

The CA-CFAR will suffer in performance when other objects are present in the leading and/or lagging reference cells. In such cases the value of the test statistic $\bar{q}$ may significantly increase, leading to missed targets. This is illustrated in Fig. 2.13 where a signal $\boldsymbol{x}$ is shown together with the adaptive threshold level $T\bar{q}$ for three different CFAR methods. We see that targets A and C both have target B in their reference cells. As a result $\bar{q}$ is increased and both targets A and C are missed when using CA-CFAR.

An alternative to CA-CFAR is smallest of CFAR (SO-CFAR) where the test statistic is instead chosen as

$$\bar{q} = \frac{1}{L} \min \left(\sum_{l=1}^{L} \boldsymbol{x}_{\text{lagg}}[l], \sum_{l=1}^{L} \boldsymbol{x}_{\text{lead}}[l] \right). \tag{2.39}$$

This will remove interference in $\bar{q}$ in cases where either the leading or lagging reference cells contain other targets, but not both. As we see in Fig. 2.13, target A is detected when using SO-CFAR since the lagging reference cells are free from interference, but target C remains undetected due to interference on both sides of the CUT.

Another CFAR approach is the ordered statistic CFAR (OS-CFAR) [47], where the test statistic $\bar{q}$ is instead chosen as the k'th largest sample from all reference cells where $k = [1, 2L]$. This will behave quite well as long as the number of cells containing interference is less than $2L - k$, regardless if these are in the leading or lagging window. This also reduces the need for guard cells.

Regardless of which CFAR filter we use, (and there are many more not mentioned here), it is the relative signal power that is considered in the detection process, not the shape of the signal pattern. Incorporating this pattern may however

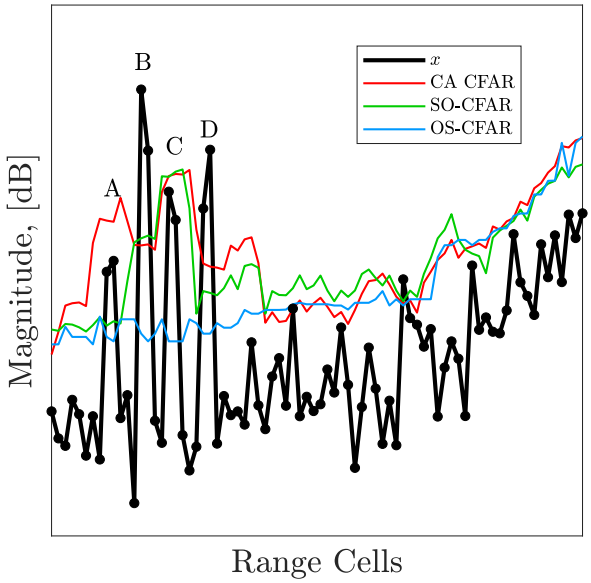


Figure 2.13: CFAR Example with four targets. The figure shows the adaptive thresholds $T\bar{q}$ for different CFAR methods. In all cases the threshold T was set to yield a false alarm probability of 1%. The number of guard cells was two on each side of the CUT and the window size for each side was $L = 5$. We note that a constant threshold across all range cells low enough to detect target A, would result in several false alarms at the far right cells.

significantly improve the trade-off between false alarms and true detections. As we see in Fig. 2.13, simple filters can fail to detect targets that to the human eye may appear as obvious. In Chapter 5 we discuss the use of ANNs for this very purpose, where we make use of temporal patterns in the detection process.



Figure 2.14: A fishing fly imitation of a wasp. There are obvious differences, or errors, between this model and an actual wasp. In spite of this, many trout have been caught by the author with this particular fly pattern.

2.7 Target Modeling and the Radar Cross Section

Before discussing target modeling in radar, we will direct our attention to fly fishing. The fly fisherman will catch fish on imitations of insects called flies. Fly fishing is an early example of the concept of modeling in general, and serves as a good reference for what we aim to achieve with the radar target models. One of the first publications on how to *model* insects dates back to 1496 [48], although there is evidence of fishing flies being used for thousands of years [49, Ch. 9]. The flies are designed to imitate insects that constitute fish's natural diet. At closer inspection however, most such flies do not actually look like the thing they are supposed to imitate, such as the wasp fly in Fig. 2.14. They are nonetheless good models in the sense that they are sufficiently similar to real insects to fool the trout. In our case of radar target modeling, our intention is not to fool the natural intelligence of fish, but rather train the artificial intelligence of neural networks.

Our motivation for using synthetic data in this manner is discussed in greater detail in Section 3.3. Here we simply note that the end goal is to obtain ANNs that evaluate similarly on "imitated" data as it would on real data. Fidelity to the underlying physics of the target model is important but not the main objective. We will therefore allow some assumptions that do not necessarily have direct physical justifications.

Fundamental to modeling target returns in radar is the RCS. The RCS, denoted here as σ, is used to calculate how strong a received echo will be from a target, or scatterer, at a particular aspect angle. Consequently, it can be used to determine

whether the object will be detectable or not. It is defined as the limit of the ratio

$$\sigma = 4\pi \frac{\text{power per unit solid angle scattered in a specified direction}}{\begin{array}{c}\text{power per unit area in a plane wave incident} \\ \text{on the scatterer from a specified direction}\end{array}} \tag{2.40}$$

when the distance from the scatterer to the point where the scattered power is measured approaches infinity [50]. Apart from aspect angle, wavelength and polarisation, the RCS will also depend on the surface shape of the target and the conductivity of the surface material. The RCS of a sphere can be calculated from Mie theory and is given in [6, Sec. 6.1]. A perfectly conducting sphere with radius a will have normalized RCS

$$\frac{\sigma}{\pi a^2} = \frac{1}{\rho^2}\left|\sum_{n=0}^{\infty}(-1)^2(2n+1)\left(a_n^s - b_n^s\right)\right|^2 \tag{2.41}$$

where we have normalized by the cross sectional area of the sphere. Here

$$\rho = \frac{2\pi a}{\lambda} \tag{2.42}$$

$$a_n^s = -\frac{j_n(\rho)}{h_n^{(2)}(\rho)} \tag{2.43}$$

$$b_n^s = -\frac{\rho j_{n-1}(\rho) - n j_n(\rho)}{\rho h_{n-1}^{(2)}(\rho) - n h_n^{(2)}(\rho)}, \tag{2.44}$$

where λ is the wavelength, $j_n(\cdot)$ is the spherical Bessel function of the first kind and $h_n^{(2)}(\cdot)$ is the spherical Hankel function of the second kind. This normalized RCS is shown in Fig. 2.15 as a function of the ratio of the radius over wavelength. For large ratios, such as $a/\lambda > 10$, the scattering enters the optical region where the RCS is simply given as

$$\sigma = \pi a^2 \tag{2.45}$$

i.e. the actual cross section of a sphere. This is independent of both wavelength and aspect angle, making it convenient for modeling targets. It can also be used to gain some intuition of the reflectivity of a target. In Chapter 6 we will see that birds have an estimated RCS of about 0.01 to 0.001 m^2 at 77 GHz (λ = 3.8 mm). In terms of signal strength, we can thus think of birds as metal sphere targets with 2 to 6 cm radius using (2.45). This analogy will however fail when we leave the optical region in Fig. 2.15.

By generating "clouds" of spherical point scatterers, we can generate arbitrary

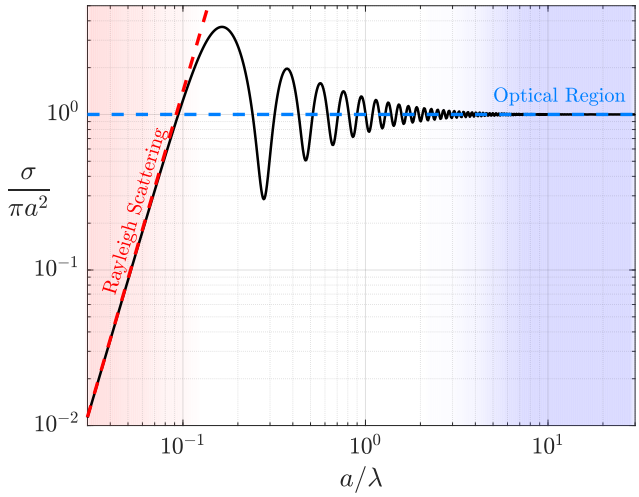


Figure 2.15: Mie scattering of a conducting sphere. In the optical region where a/λ is large, the RCS becomes independent of wavelength. For small ratios, the RCS is given as $\sigma = \pi a^2 9\rho^4 = 144\pi^5 a^6/\lambda^4$, known as Rayleigh Scattering.

targets. Let such a two dimensional cloud consist of N scatterers, each with polar coordinates (ϕ_n, ρ_n) and RCS σ_n, for $n = 1, ..., N_s$. Ignoring secondary reflections between scatterers, the total angular dependent RCS can be modeled as

$$\sigma(\theta) = \left| \sum_{n=1}^{N_s} \sqrt{\sigma_n} e^{j\frac{2\pi\rho_n}{\lambda}\sin(\theta-\phi_n)} \right|^2 . \tag{2.46}$$

In Fig. 2.16 we see examples of such randomly sampled clouds where the radius coordinate is limited to $\rho_n \leq r_s$, and the total RCS is plotted in decibels relative to one square meter (dBsm) as a function of angle. Fig. 2.16a - 2.16c are obtained from (2.46). Fig. 2.16d however, is from an actual RCS measurement of a drone. We can make the following observations

1. Without much effort we have created an RCS model that is similar to actual measurements, at least when comparing Fig. 2.16c and Fig. 2.16d.

2. The variations in RCS is a reflection of the target shape, and we thus have reason to believe that the target geometry could to some extent be inferred from the RCS pattern. This is in fact the premise to our study in Chapter 4, where we instead of illuminating the target at different aspect

angles, illuminate the target at different frequencies.

3. Even if we cannot perfectly, or unambiguously, describe the target's geometry from an RCS pattern, it could still be used for distinguishing between different types of targets. Suppose we had a function that could estimate N_s and r_s based on the model in (2.46). If evaluated on a true target such as the drone in 2.16d, we may assume that the estimates of these parameters would more likely have values closer to $N_s = 40$, and $r_s = 40\lambda$ than it would to the other two examples in Fig. 2.16a and 2.16b. Note that for classification purposes, the estimated parameters do not even need to have a physical interpretation. This is a premise to our study in Chapter 6, where we instead look at models of the micro-Doppler spectrum.

4. The RCS pattern appears to become more random for larger N_s and r_s.

Regarding point 3 in the above list, we note that the model in (2.46) will not be a good match to micro-Doppler spectra as it does not model any time dependent dynamics. For this purpose we will also use the analytical model for a metal cylinder. We "rotate" this cylinder to model propeller blades on drones, as well as moving parts on humans and birds. See Chapter 6. The RCS of a cylinder is given in [3, Ch. 3] as

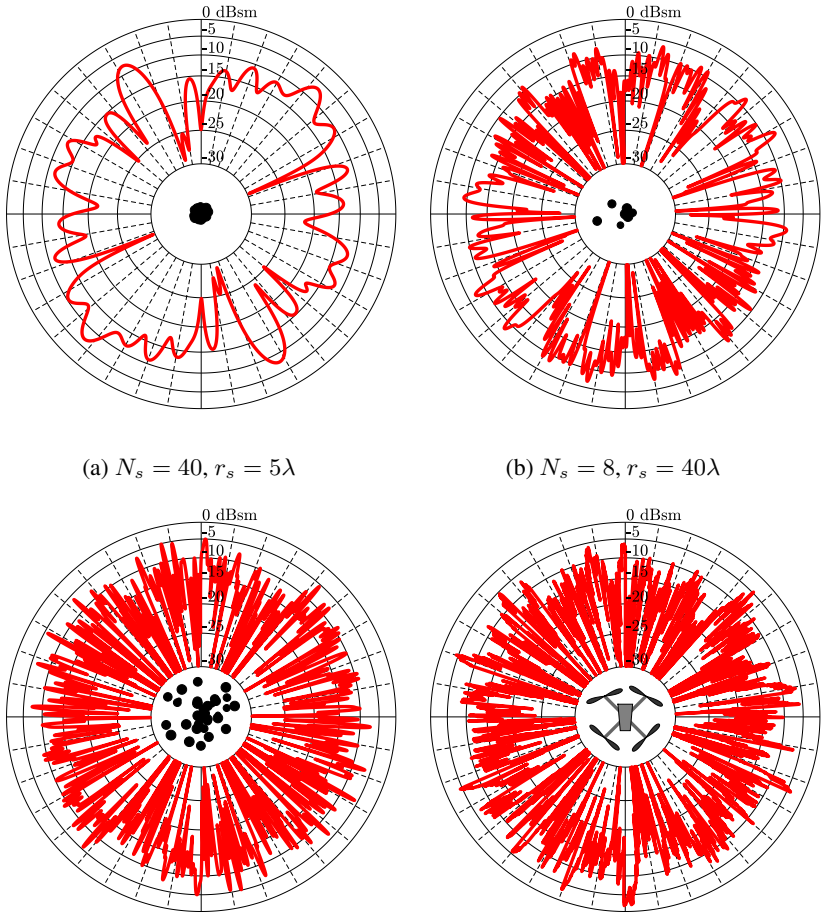
$$\begin{aligned}\sigma(\theta) &= \sigma_0 \left[\frac{\sin\left(2\pi L \sin(\theta)/\lambda\right)}{2\pi L \sin(\theta)/\lambda}\right]^2 \cos^2(\theta) \\ &\approx \sigma_0 \left[\frac{\sin\left(2\pi L \sin(\theta)/\lambda\right)}{2\pi L \sin(\theta)/\lambda}\right]^2 = \sigma_0 \text{sinc}^2\left(2\pi L \sin(\theta)/\lambda\right) \qquad (2.47)\end{aligned}$$

$$\sigma_0 = \frac{2\pi a L^2}{\lambda} \qquad (2.48)$$

where L is the cylinder length and a is the radius, see Fig. 2.17. In Chapter 6 we present an RCS model of drones based on clusters of rotating cylinders and point scatterers, and compare this to real measurements at 77 GHz.

Regarding point 4 in the above list, one can expect the RCS fluctuations to be seemingly random for even slight variations in aspect angle when N_s and r_s are large. This is the underlying assumption for the four Swerling models presented in [30]. It can be shown that for an infinitely large number of point scatterers, the distribution of the RCS at an arbitrary angle will have a chi-square PDF with two degrees of freedom

$$p(\sigma) = \frac{1}{\bar{\sigma}} e^{-\frac{\sigma}{\bar{\sigma}}} \qquad (2.49)$$



(a) $N_s = 40$, $r_s = 5\lambda$

(b) $N_s = 8$, $r_s = 40\lambda$

(c) $N_s = 40$, $r_s = 40\lambda$

(d) DJI Mavic 2 Pro at 77 GHz

Figure 2.16: RCS in dBsm as a function of aspect angle. Image (a)-(c) are examples of simulated targets generated as uniformly sampled clouds of scatterers. Image (d) is an actual RCS measurement of a drone, measured at 77 GHz. The drone was hovering and rotating $360°$ during the measurement. Much of the variations in the RCS is due to the rotation of the propellers. The scatterer RCS in(a)-(c) was uniformly sampled in $[0, 1]$ and normalized to have the same mean RCS as the drone in (d). Note that unlike image (d), the simulated targets in (a)-(c) have symmetric RCS.

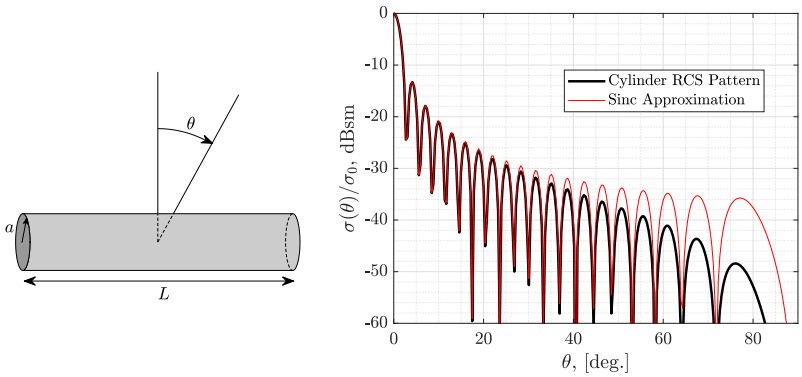


Figure 2.17: The RCS pattern of a cylinder assuming that the dimensions of a and L are larger than the wavelength. This model assumes an infinitely thin cylinder pipe and does not incorporate reflections from the ends at $\theta = \pm 90°$. At these angles, a solid cylinder would have an RCS of $4\pi A^2/\lambda^2$, where $A = \pi a^2$.

where $\bar{\sigma}$ is the average RCS. If we assume that the RCS can be regarded as independent samples of (2.49) between scans, we obatin the Swerling I model. A Swerling II model assumes that the RCS fluctuates with a shorter correlation time, where the RCS samples are instead modeled as independent between pulses. Swerling III and IV targets are similar, but the target is instead modeled as a cloud of scatterers with a single larger, or dominant, scatterer. The PDF of the RCS of such targets is a chi-square distribution with four degrees of freedom

$$p(\sigma) = \frac{4\sigma}{\bar{\sigma}^2} e^{-\frac{2\sigma}{\bar{\sigma}}}. \tag{2.50}$$

Swerling III targets are modeled as independent samples of (2.50) between scans and Swerling IV targets as independent samples of (2.50) between pulses. A target that does not fluctuate in RCS is sometimes referred to as a Swerling 0 target, or a Marcum model [29].

2.8 Sidelobes and Windowing

In radar, the dynamic range in target power may be quite large, and the power value from weaker targets may then be covered by sidelobes from significantly stronger targets. Such sidelobe masking or interference may appear in angle, range and velocity processing. The sidelobe levels can be reduced using window functions [51], where the elements are weighted before transform to angle, range or velocity domain.

In this thesis we will use the Taylor window [52] for this purpose. The N-point Taylor window is given as

$$\boldsymbol{w}[n] = 1 + 2\sum_{m=1}^{\bar{n}-1} F_m \cos\left(2\pi m\left(\frac{n-1}{N-1} - \frac{1}{2}\right)\right) \tag{2.51}$$

for $n = 1, ..., N$, where

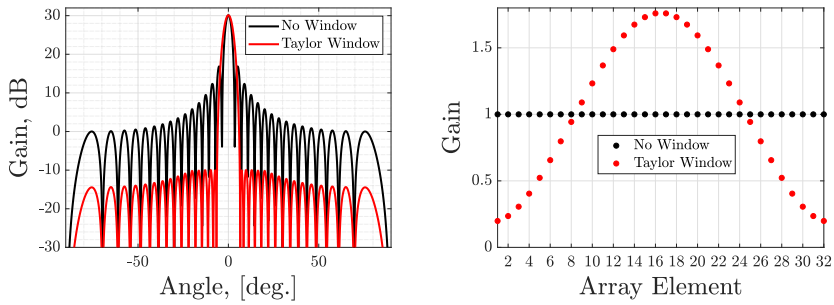
$$F_m = -\frac{(-1)^m}{2}\frac{\prod_{l=1}^{\bar{n}-1}\left(1 - \frac{m^2}{\bar{n}^2}\frac{A^2+\left(\bar{n}-\frac{1}{2}\right)^2}{A^2+\left(l-\frac{1}{2}\right)^2}\right)}{\prod_{l=1,l\neq m}^{\bar{n}-1}\left(1-\frac{m^2}{l^2}\right)} \tag{2.52}$$

$$A = \frac{\log\left(\eta + \sqrt{\eta^2-1}\right)}{\pi} \tag{2.53}$$

$$\eta = 10^{-S/20} \tag{2.54}$$

where the parameter $S < 0$ is the resulting sidelobe suppression level in dB, and the parameter $\bar{n}$ is the number of sidelobes adjacent to the main lobe with approximately constant level. The desired sidelobe characteristics set by S and $\bar{n}$ are however not always achievable, and S and $\bar{n}$ should therefore be limited. According to [51], such limitations are $|S|/9 < \bar{n} < N/5$ for $-50 < S < -30$.

An example of a 32-point Taylor window on a ULA is shown in Fig. 2.18. We note that the sidelobe reduction comes at a cost of increased beamwidth, in this case by a factor 1.4. The element gains or weights are shown in Fig. 2.18b. Both sets of weights sum to 32 yielding equal total gain. In practical systems however, this will require individual array elements to reduce their gain. Windowing is therefore also associated with a loss in total gain. Allowing a maximum element gain of one in Fig. 2.18b, would reduce the gain in Fig. 2.18b by 4.8 dB in the Taylor window case.



(a) Resulting antenna gain

(b) Window weights.

Figure 2.18: Example of a Taylor window on a 32 element ULA. The Taylor window parameters are $S = -40$ and $\bar{n} = 5$. We see in (a) that the five sidelobe peaks on both sides of the main beam are approximately 40 dB below the main beam peak.

2.9 Frequency Modulated Continuous Wave

Our study in Chapter 6 involves an FMCW radar and we will therefore present this technology in more detail. In FMCW radar, the frequency of the CW is continuously changed, typically implemented as a linear change with sawtooth or triangular pattern. Although this frequency modulation can also be nonlinear we will focus on the linear case. Mixing such a signal with a time delayed version of itself, results in a sinusoidal signal with a beat frequency that is proportional to the time delay. This is illustrated in Fig. 2.19.

Let the transmitted signal during a period T_{sweep} be linearly modulated in frequency with chirp rate

$$\alpha = \frac{f_1 - f_0}{T_{\text{sweep}}} = \frac{\Delta f}{T_{\text{sweep}}}, \tag{2.55}$$

where we sweep from frequency f_0 to f_1. We then have

$$\begin{aligned} s(t) &= \sin\left(2\pi\left[\int_{u=0}^{t}(\alpha u + f_0)\,du\right] + \phi\right) \\ &= \sin\left(2\pi\left[\frac{\alpha}{2}t^2 + f_0 t\right] + \phi\right) \end{aligned} \tag{2.56}$$

for $0 \leq t \leq T_{\text{sweep}}$, and some arbitrary phase ϕ. For a sawtooth pattern this chirp is then repeated with intervals of T_{sweep}, known as FM-sweeps. In a triangular pattern,

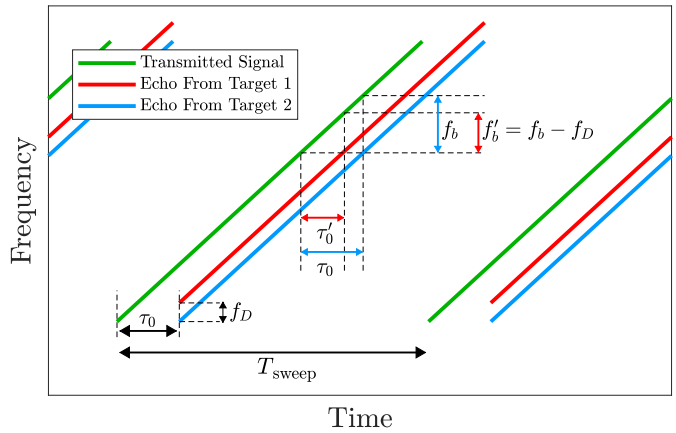


Figure 2.19: A two target FMCW example. The plot shows the linear frequency modulation for the transmitted and received signals. Two targets are located at the same range, corresponding to time delay τ_0. Target 1 is in motion, moving towards the radar. The echo from Target 1 is therefore Doppler shifted in frequency by $f_D > 0$. The (constant) frequency difference between the transmitted wave and echo 1 is then reduced to $f_b' = f_b - f_D$ where f_b is the frequency difference between the transmitted wave and echo 2. The reduced frequency will correspond to a reduced time delay $\tau_0' < \tau_0$, and Target 1 will appear closer to the radar than the stationary Target 2.

the chirp rate becomes negative in the next FM-sweep, sweeping back in frequency from f_1 down to f_0, at which point the waveform is repeated and so forth.

Using the Doppler time scaling factor β in (2.12), we can express the received reflection from a moving target at delay τ_0 as

$$\begin{aligned} s'(t) \propto \tilde{s}(t-\tau_0) &= \sin\left(2\pi\left[\int_{u=0}^{t-\tau_0}(\alpha u + f_0)\,\beta du\right] + \phi\right) \\ &= \sin\left(2\pi\beta\left[\frac{\alpha}{2}(t-\tau_0)^2 + f_0(t-\tau_0)\right] + \phi\right), \end{aligned} \tag{2.57}$$

and the mixed signal at the receiver as

$$x(t) = s(t)s'(t) \propto s(t)\tilde{s}(t-\tau_0). \tag{2.58}$$

From (2.56) and (2.57), let

$$A = 2\pi \left[\frac{\alpha}{2}t^2 + f_0 t\right] + \phi \tag{2.59}$$

and

$$B = 2\pi\beta \left[\frac{\alpha}{2}\left(t - \tau_0\right)^2 + f_0(t - \tau_0)\right] + \phi. \tag{2.60}$$

be the arguments of the two mixed signals in (2.58). Using the trigonometric identity

$$2\sin(A)\sin(B) = \cos(A - B) - \cos(A + B), \tag{2.61}$$

we note that the term $A + B$ will have a frequency on the order of $2f_0$. With a sufficient low pass filter, this component can be ignored. The term $A - B$ can be written as

$$A - B = 2\pi \left[f_b + f'_D\right] t + \varphi \tag{2.62}$$

where

$$f_b = \beta\alpha\tau_0 \approx \alpha\tau = \frac{\Delta f}{T_{\text{sweep}}}\tau_0 \tag{2.63}$$

$$f'_D = -\frac{2v_r}{c}f_0 - \frac{\alpha v_r}{c}t \tag{2.64}$$

$$\varphi = 2\pi\beta \left[f_0 - \frac{\alpha}{2}\right] \tau_0 \approx 2\pi \left[f_0 - \frac{\alpha}{2}\right] \tau_0. \tag{2.65}$$

The Doppler frequency in (2.64) varies slightly with time. An average frequency can be obtained by taking the mean of the time derivative of $f'_D t$. Since this is linear we take

$$\begin{aligned}
\left(\frac{\partial}{\partial t} f'_D t\right)\Big|_{t=T_{\text{sweep}}/2} &= \left(\frac{\partial}{\partial t}\left[-\frac{2v_r}{c}f_0 t - \frac{\alpha v_r}{c}t^2\right]\right)\Big|_{t=T_{\text{sweep}}/2} \\
&= \left[-\frac{2v_r}{c}f_0 - \alpha\frac{2v_r}{c}t\right]\Big|_{t=T_{\text{sweep}}/2} \\
&= -\frac{2v_r}{c}\left(f_0 + \frac{\Delta f}{2}\right) = -\frac{2v_r}{c}f_c = -f_D,
\end{aligned} \tag{2.66}$$

where we used (2.55) in the last step, and f_c is the center frequency between f_0 and f_1. Note that the sign is opposite from the Doppler frequency we saw in (2.14). This

finally gives us

$$x(t) \propto \sin\left(2\pi\left[f_b - f_D\right]t + \phi\right). \tag{2.67}$$

The relation between f_D, f_b and the time delay τ_0 is illustrated in Fig. 2.19.

For stationary cases where $v_r = f_D = 0$, the range to the target can be estimated from the beat frequency f_b in (2.63) as

$$r = \frac{c\tau_0}{2} = \frac{cf_bT_{\text{sweep}}}{2\Delta f}. \tag{2.68}$$

Regarding the highest resolution achievable with this type of system, we first note that the magnitude of the Fourier transform of (2.67) will be the magnitude of the sum of two sinc functions centred at $\pm(f_b - f_D)$. Each such sinc function,

$$\text{sinc}\left(fT_{\text{sweep}}\right) = \frac{\sin\left(\pi fT_{\text{sweep}}\right)}{\pi fT_{\text{sweep}}}, \tag{2.69}$$

has a 3 dB resolution of $\partial f = 0.886/T_{\text{sweep}}$ Hz. A more convenient metric is the 4 dB resolution at $\partial f = 1/T_{\text{sweep}}$ Hz, which put into (2.68), yields the range resolution

$$\partial r = \frac{c\partial fT_{\text{sweep}}}{2\Delta f} = \frac{c}{2\Delta f}. \tag{2.70}$$

Noting that Δf is the bandwidth, this is consistent with the expression for the range resolution we saw in (2.3).

In practice, $x(t)$ will be a sum of several beat frequencies coming from several targets. Peaks at certain frequencies in the spectrum of $x(t)$ then correspond to targets at certain ranges. This spectrum is typically estimated with the DFT of the sampled version of $x(t)$, and is efficiently computed with the fast Fourier transform (FFT) algorithm. This process of matching $x(t)$, or $x[nT_s]$ in discrete time, to a sinusoidal signal is known as range compression and is analogue to pulse compression in pulsed radar systems. After sampling, the practical range ambiguity will not only be limited by the PRF, or $1/T_{sweep}$, but also by the sampling rate. For real valued signals sampled at f_s Hz, the Nyquist frequency $f_b^{\max} = f_s/2$ corresponds to the maximum range before aliasing occurs.

When $f_D \neq 0$, the estimated beat frequency will be the sum of the actual beat frequency f_b and the Doppler frequency $-f_D$. This ambiguity cannot be resolved in a single sweep. If a triangular waveform is used, the Doppler frequency will have opposite sign when the chirp is negative, and f_b and f_D can then be resolved from two consecutive sweeps. For sawtooth shaped waveforms, the Doppler frequency

Table 2.2: FMCW Radar Specifications

Type	SAAB SIRS 1600
Center Frequency	77 GHz
Bandwidth	160 MHz
Range Resolution	1 m
Range Sampling Resolution	0.943 m
PRF	17 kHz
Azimuth Beamwidth	1°
Scan Rate	10 Hz, mechanical
Field of View	$\pm 9^\circ$
Output Power	10 mW
Wave Pattern	Sawtooth

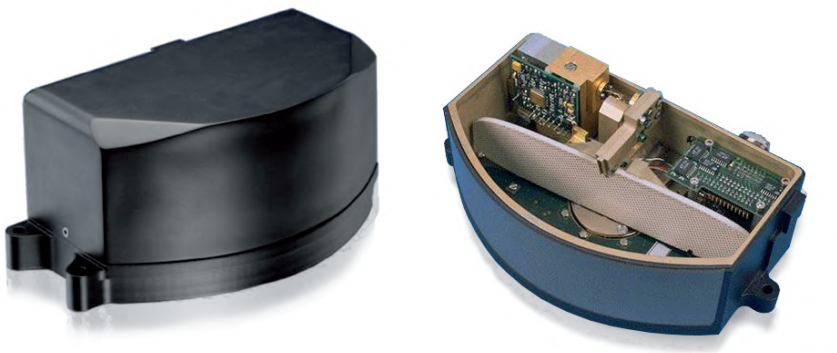


Figure 2.20: The Saab SIRS 1600 FMCW radar.

can be estimated from a series of FM-sweeps (slow time processing) in the manner described in Section 2.4. Depending on scenario the error caused by f_D may or may not be significant. As an example, in Fig. 2.5 we can read out a velocity of -8 m/s for the drone. Using (2.66), this corresponds to a frequency offset of 4.1 kHz, which in turn yields a range offset of 16 cm using (2.68). This is only a fraction of the range resolution of 1 m.

The FMCW sensor used in this thesis is the Saab manufactured millimeter-wave radar (mmWR) shown in Fig. 2.20. It is a mechanically scanning radar. The dwell time, i.e. the time a target is illuminated by the radar in a processing interval, is limited to the scan rate and beamwidth, resulting in 4 ms for this sensor (using the 3dB beamwidth). The sensor parameters are listed in Table 2.2.

Chapter 3

Background in Artificial Neural Networks

In recent years, the field of AI has seen much progress, and consequently an increase in interest. Although such interest has sprung up and faded away several times since the 1950s, current technology based on ANNs and deep learning, is now being used to such an extent that its long term relevance is difficult to question.

In this chapter we present some background in machine learning with ANNs. The usage of ANNs is closely linked to the more general field of AI. We may define AI in numerous ways depending on viewpoint [53, Ch. 1]. The following text by Frank Rosenblatt from 1957 [54, p. 1] serves as a good description of the objective of AI from an engineering perspective:

> Since the advent of electronic computers and modern servo systems, an increasing amount of attention has been focused on the feasibility of constructing a device possessing such human-like functions as perception, recognition, concept formation, and the ability to generalise from experience. In particular, interest has centered on the idea of a machine which would be capable of conceptualising inputs impinging directly from the physical environment of light, sound, temperature etc. – the "phenomenal world" with which we are all familiar – rather than requiring the intervention of a human agent to digest and code the necessary information.

ANNs and the methods for training these, have been developed for the purpose of solving such tasks. It should however be noted that the "intervention of a human agent" is still a critical part of the current state of the art, even if this intervention is more focused on the training process rather than "digesting" the information in the data.

In Section 3.1 we present simple forms of ANNs. In Section 3.2 we expand this to other types of architectures used in this thesis, and in Section 3.3 we discuss the training methods used for optimising the ANNs in Chapters 4 - 7.

3.1 Artificial Neural Networks

As the name implies, ANNs are inspired by the biological nervous system, where cells called neurons are connected in large networks, e.g. an estimated 85 billion neurons in a human brain [55]. Simply described, these neurons receive stimuli, or to use signal processing terminology "input signals". Depending on the input, the neuron may relay a new signal(s) to other neurons, an "output signal". With this simplified notion of a biological neuron, the first mathematical model of such a cell was published in 1943 [56]. It was binary valued and had active output if the number of active inputs exceeded a set threshold.

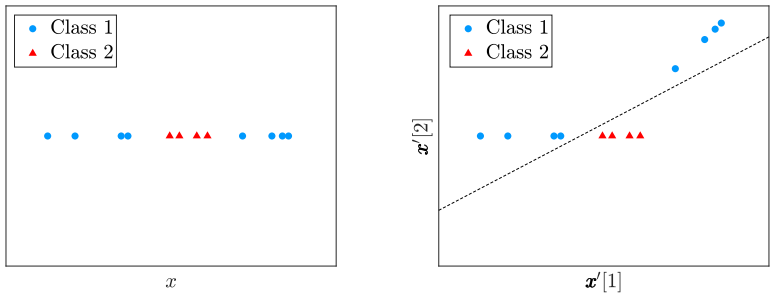
This single neuron model was later refined by Frank Rosenblatt in 1957, with the ground breaking perceptron [54], [57]. In addition to the previous models, the perceptron arbitrarily weighted each input before a final summation, and could be trained via various "reinforcement systems" [58, Sec. 4.6]. Its ability to distinguish between different types of patterns was successfully demonstrated in both computer simulations [59], and with a hardware implementation, where a machine named the "Mark I" perceptron, could successfully classify different patterns such as letters [60]. The perception can be mathematically expressed as

$$\hat{y} = a\left(\boldsymbol{w}^T\boldsymbol{x} + b\right) \tag{3.1}$$

$$a(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases} \tag{3.2}$$

where $\boldsymbol{w} \in \mathbb{R}^{D\times 1}$ are the weights, b is a bias, and the elements in $\boldsymbol{x} \in \mathbb{R}^{D\times 1}$ are each a function of the input, referred to as an association unit. In the case of the Mark I perceptron, this was a number of randomly connected pixels from a 20×20 pixel input image. The function $a(\cdot)$ has later become known as an activation function in the field of machine learning.

The training of the Mark I perceptron did not depend on the input data but came from optimising a cost function to which the correct predictions, or labels y, were initially given. In this optimisation process the weights, implemented with potentiometers, were mechanically tuned via electric motors [61, Sec. 4.1]. For linearly separable classification problems, it can be shown that such a classifier will, after a sufficient amount of training iterations, converge to a perfect solution [58, Sec. 5], [61, Sec. 4.1]. A perfect solution in this case implies that $y_i\hat{y}_i = 1\ \forall\, i$. For



(a) (b)

Figure 3.1: Example of nonlinearity in binary classification. The original data in (a) is one dimensional, and the two classes are not linearly separable. In (b) they become linearly separable after introducing a second dimension with the nonlinear function in (3.3).

classification problems with multiple classes, several perceptrons were added in parallel to form an output $\hat{\boldsymbol{y}} \in \mathbb{R}^{K\times 1}$. The dimension K of the output will depend on the encoding of the classes. When training the Mark I perceptron on all 26 letters of the alphabet, a five bit code was used, i.e. $K = 5$ [60]. A perhaps more common encoding in modern implementations is the "one hot" representation, where K is equal to the total number of classes and the label vector $\boldsymbol{y}$ is non-zero valued at one element only. We will come back to this when discussing losses in Section 3.3.1.

This single layer perceptron will however fail when the problem is not linearly separable. This can, for certain datasets, be overcome by introducing nonlinearities. As an example, consider the binary classification problem in Fig. 3.1. The problem becomes linearly separable when using the nonlinear (but piecewise linear) function

$$\boldsymbol{x}' = a\left(\begin{bmatrix}1\\ w\end{bmatrix} x + \begin{bmatrix}b_1\\ b_2\end{bmatrix}\right) = a(\boldsymbol{W}x + \boldsymbol{b}) \tag{3.3}$$

$$a(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases} \tag{3.4}$$

where the rectified linear unit (ReLu) function $a(\cdot)$, is scalar valued in the sense that it operates on each element in the input. Here b_1 ensures that $x+b_1$ is positive for all samples, and b_2 ensures that $wx + b_2$ is greater than zero only for the four samples to the far right in Fig. 3.1b. We note that the problem becomes separable for many

possible values of w, and for each such value there are many possible separating hyperplanes (dashed boundary line in Fig. 3.1b). When comparing (3.3) and (3.4) with (3.1) and (3.2), we note that (3.3) is two perceptrons in parallel but with a different activation function. Although the original formulation of the perceptron had the threshold activation in (3.2), the term perceptron is often used with any type of nonlinear activation function, even if the term artificial neuron would in such cases be more correct. For the sake of simplicity, we will not make any distinction between these in this thesis.

The data samples in Fig. 3.1 can now be perfectly classified by feeding $\boldsymbol{x}'$ as input to a second layer with a single perceptron. In general, several layers of perceptrons with nonlinear activations can be added to form more complex decision boundaries. Except for the last layer, the dimensionality of each layer, i.e. the number of perceptrons in each layer, can be arbitrary. The output from each such perception is referred to as a node. Such multilayer perceptrons (MLPs) are present in some form in most ANNs.

3.1.1 Universal Function Approximators

We have now introduced the MLP as a classifier. We can also regard a MLP as an arbitrary nonlinear function, with many degrees of freedom, or high complexity. In fact, is has been shown that a MLP with a single hidden layer, see Fig 3.2, can approximate any function [62]. This may however require an infinite dimensionality of the hidden layer. The high dimensionality requirement can be worked around by concatenating several layers, forming a deep neural network. As an example, consider the polynomial consisting of the product of all elements in a vector

$$y = \prod_{i=1}^{N} \boldsymbol{x}[i]. \tag{3.5}$$

It has been shown that for even N, (3.5) can be evaluated by a deep neural network with a total of $4N$ neurons in $\log_2(N)$ layers. In contrast, a network with a single hidden layer would require 2^N neurons [63].

Regarding the expressivity of a MLP, we can gain some intuition by noting the following. Let $\boldsymbol{W} \in \mathbb{R}^{D_1 \times D_0}$ be a weight matrix and $a(\cdot)$ be the ReLu function in (3.4). Then, for a particular input $\boldsymbol{x}$, we have that

$$\boldsymbol{x}' = a\left(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}\right) = \tilde{\boldsymbol{W}}\boldsymbol{x} + \tilde{\boldsymbol{b}} \tag{3.6}$$

where the rows of $\tilde{\boldsymbol{W}} \in \mathbb{R}^{D_1 \times D_0}$ and $\tilde{\boldsymbol{b}} \in \mathbb{R}^{D_1 \times 1}$ are either zero valued or have the same values as the corresponding row in $\boldsymbol{W}$ and $\boldsymbol{b}$. Since each row can take one of two forms, the function in (3.6) can thus represent up to 2^{D_1} different affine

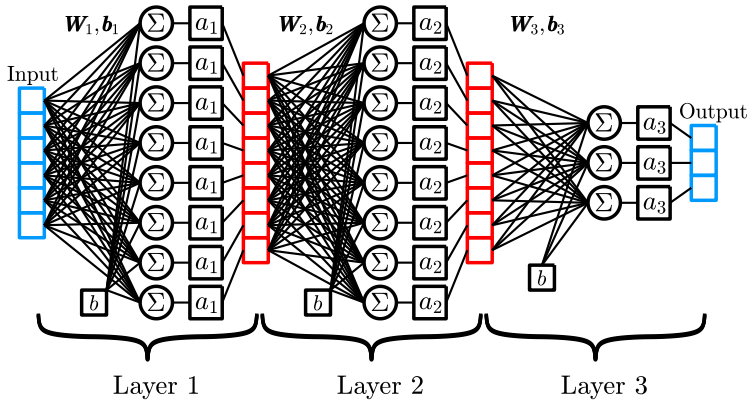


Figure 3.2: Illustration of a three layer MLP architecture with eight nodes in layers one and two, and three nodes in the final output layer. The output $\boldsymbol{y}$ is related to the input $\boldsymbol{x}$ as $\boldsymbol{y} = a_3\left(\boldsymbol{W}_3 a_2\left(\boldsymbol{W}_2 a_1\left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1\right) + \boldsymbol{b}_2\right) + \boldsymbol{b}_3\right)$. Layer 1 and layer 2 are referred to as hidden layers since the outputs from these, (colored in red) are not observed.

transforms, depending on the input. This is an upper bound and it is not difficult to show examples of partitions that cannot be attained, especially when $D_1 = 1$. A tighter bound is given in [64] as

$$M \leq \sum_{j=0}^{D_0} \binom{D_1}{j} \leq 2^{D_1} \tag{3.7}$$

where M is the number of linear regions that the input space has been partitioned into. A ReLu activated MLP network having K layers with D_k nodes in each layer, can thus represent at most

$$M \leq \prod_{d=1}^{K} \sum_{j=0}^{D_{k-1}} \binom{D_k}{j} \tag{3.8}$$

different affine transforms. Although this bound can be made slightly tighter, [65, 66], we will end our discussion on this matter by noting that M in (3.8) quickly becomes very large for even moderate values of K and $D_0, ..., D_K$. This potential expressivity, or complexity, of such ANNs gives some intuition as to why we can regard ANNs as universal function approximators.

It is however not obvious if all these affine transforms are attainable, or if they are much different. Neither is it trivial to set the parameters in all the layers such that the desired function is obtained, even if we happen to know from theory that it can be formed. How to train MLPs and ANNs in general is discussed in Section 3.3.

3.2 Architectures

We introduced the MLP in Section 3.1. In this section we present the convolutional neural network (CNN) and the autoencoder implemented with ANNs.

3.2.1 Convolutional Neural Networks

A layer in an ANN may have different types of connections between input and output nodes. In MLPs, all layers are fully connected, where all input nodes are connected to all output nodes. Another type of layer used in this thesis is the convolution layer. An ANN that has convolution layers is often referred to as a CNN, even if not all layers are convolutional.

The concept of layers with convolutions was first introduced in [67] in 1980. However, the structure we will now introduce with multiple filters in each layer, originates from the "LeNet" structure in [68] from 1998. In such layers, only a local segment in the input nodes contributes to an output node via an artificial neuron. The neuron is then convolved over the entire input. This may be repeated with multiple artificial neurons filters, producing an output from each such neuron in different channels. This is illustrated with an example in Fig. 3.3 for a one dimensional input. A convolution can be performed in higher dimensions. In this thesis we will consider at most two.

Let the input to a convolutional layer be $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2 \times D}$ where D is the number of channels. If the input is an image, then it is often the case that $D = 3$, for the red green blue (RGB) color coordinate values of each pixel. If the input corresponds to an output from a previous convolutional layer, then D is the number of filters in the previous layer. Note that the vector input in Fig. 3.3 is just a special case of this where $D = N_2 = 1$. Let $\boldsymbol{F}_k \in \mathbb{R}^{k_1 \times k_2 \times D}$ and b_k, be the weights and a bias of the k'th filter in a layer for $k = 1, ..., K$. This has segment size, or kernel size, $k_1 \times k_2$. Considering a single layer, the output from the convolution of this

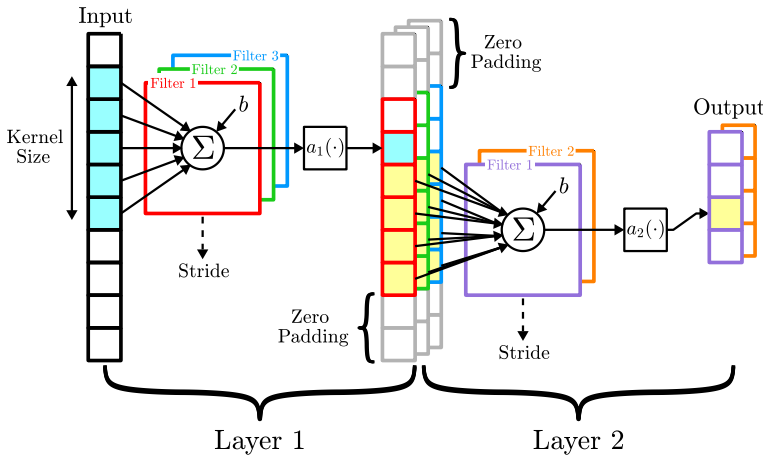


Figure 3.3: Illustration of a two layer CNN architecture. In layer one, the input with size 10×1 is filtered by three filters with kernel size $k_1 = 5$, and stride $s_1 = 1$. This produces one output of size 6×1 for each filter, stored in three channels. In layer two, the convolution includes all channel elements. The input to layer two is in this case zero padded with $z_1 = 2$. The kernel size is $k_1 = 4$ with stride $s_1 = 2$, yielding a 4×1 output for each filter.

artificial neuron filter is then $\boldsymbol{X}' \in \mathbb{R}^{N_1' \times N_2' \times K}$ with elements

$$\boldsymbol{X}'[n_1, n_2, k] = a\left(\boldsymbol{Z}[n_1, n_2, k] + b_k\right) \tag{3.9}$$

$$\boldsymbol{Z}[n_1, n_2, k] = \sum_{d=1}^{D} \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} \boldsymbol{X}[(n_1 - 1)s_1 + i, (n_2 - 1)s_2 + j, d]\boldsymbol{F}_k[i, j, d] \tag{3.10}$$

where $a(\cdot)$ is some activation function, $N_i' = \lfloor (N_i - k_i)/s_i \rfloor + 1$ for $i = 1, 2$, and $\lfloor \cdot \rfloor$ denotes the floor function. The parameters s_1 and s_2, known as strides, determine the step size in the convolution. This differs from the conventional convolution seen in signal processing where the step size is typically one sample.

It is in some cases desirable to have the same input and output size which can be achieved by framing the input with zero valued elements, known as zero padding. The size of the zero padding can of course be arbitrary, and the output size in general

is

$$N_i' = \left\lfloor \frac{N_i - k_i + 2z_i}{s_i} \right\rfloor + 1 \tag{3.11}$$

for $i = 1, 2$, where z_i is the number of zeros added to each edge (beginning/end) of the input. This is illustrated in Fig 3.3 where Layer 2 is zero padded.

Another type of layer seen in CNNs is pooling layers, where arbitrary operations may be performed on the $k_1 \times k_2$ segment. An example is taking the maximum of all elements known as max pooling. Although we will not use these types of layers in this thesis, we will in Chapter 5 use a similar variant where we instead take the maximum over all channels. We will refer to this as a max channel activation. Using the same notations as in (3.9) and (3.10), this can be written as

$$\boldsymbol{X}'[n_1, n_2] = \max_k \boldsymbol{Z}[n_1, n_2, k] + b_k. \tag{3.12}$$

After each max channel activation, the number of channels is one, thereby reducing the total dimensionality, and consequently the number of required multiplications.

Much can be said about the advantages of having convolutional structures in ANNs. Here we will simply note that it significantly reduces the number of parameters to train while still achieving high dimensions in the hidden layers. Let $\boldsymbol{x} \in \mathbb{R}^{N_1 N_2 D \times 1}$ be a vector that contains all the elements of $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2 \times D}$ and let $\boldsymbol{x'} \in \mathbb{R}^{N_1' N_2' K \times 1}$ be a vector that contains all the elements of $\boldsymbol{X'} \in \mathbb{R}^{N_1' \times N_2' \times K}$. We can then express (3.9) and (3.10) as

$$\boldsymbol{x'} = a\left(\boldsymbol{W}_c \boldsymbol{x} + \boldsymbol{b}_c\right) \tag{3.13}$$

where the non-zero elements of the convolution matrix $\boldsymbol{W}_c$ are in a diagonal structure and consist of repetitions of the parameters in $\boldsymbol{F}_k$ for $k = 1, ..., K$. Likewise, the vector $\boldsymbol{b}_c$ contains repetitions of the parameters b_k for $k = 1, ..., K$. We can thus regard a convolutional layer as a special case of a fully connected layer but with many shared parameter values in $\boldsymbol{W}_c$ and $\boldsymbol{b}_c$, and many zero valued connections. In both cases the output size is $N_1' N_2' K$ whereas the number of parameters differ significantly with $K k_1 k_2 D$ parameters plus K bias parameters in a convolutional layer and $N_1' N_2' K N_1 N_2 D$ parameters plus $N_1' N_2' K$ bias parameters in a fully connected version.

As for the difference in performance, the CNN will have fewer parameters and we can thus expect it to perform worse than the MLP. It is difficult to say more than this without considering the dataset. As an example, we use the Modified National Institute of Standards and Technology (MNIST) dataset [69] for classifying digits 0-9. The dataset consists of 70,000 examples of handwritten digits with 60,000

Table 3.1: Fully Connected and Convolutional Networks

		MLP		
Layer Type	Weight Shape	Strides	Output Shape	Activation
Fully Connected	4608×784	-	4608×1	ReLu
Fully Connected	800×4608	-	800×1	ReLu
Fully Connected	72×800	-	72×1	ReLu
Fully Connected	32×72	-	32×1	ReLu
Fully Connected	32×32	-	32×1	ReLu
Fully Connected	10×32	-	10×1	Softmax
		CNN		
Layer Type	Weight Shape	Strides	Output Shape	Activation
Convolutional	F: 8, K: 5×5	[1,1]	$24 \times 24 \times 8$	ReLu
Convolutional	F: 8, K: 5×5	[2,2]	$10 \times 10 \times 8$	ReLu
Convolutional	F: 8, K: 5×5	[2,2]	$3 \times 3 \times 8$	ReLu
Fully Connected	32×72	-	32×1	ReLu
Fully Connected	32×32	-	32×1	ReLu
Fully Connected	10×32	-	10×1	Softmax

F: Number of filters/kernels. K: Kernel size. Total number of parameters in MLP: 7360320, CNN: 7048. No bias parameters in either network.

samples in the training set and 10,000 samples in the test/validation set, see Section 3.3.6. The samples are stored as 28×28 pixel images with values limited to the interval [0, 1]. To classify these we train the simple CNN and MLP in Table 3.1. Both the CNN network and the fully connected MLP have the same number of output nodes in all layers. Note that when a convolutional layer is followed by a fully connected layer, all output nodes from the convolutional layer are concatenated to a single vector. As labels, we use one hot encoding yielding an output size of 10. As described in Section 3.3, we train the two networks in 50 epochs using the categorical cross entropy loss.

The MLP network achieved 98.5% test accuracy while the CNN reached 97.7%. It is debatable whether this difference is significant. The difference in the number of parameters is however clearly significant with 7,360,320 parameters in the MLP and 7,048 parameters in the CNN. In this case no bias parameters were used, and the number of multiplications required for evaluating the MLP thus corresponds to the number of parameters. Evaluating the CNN requires 293,248 multiplications. The CNN is thus significantly cheaper to evaluate, store and train.

Table 3.2: Autoencoder using ANNs

Encoder Network, $\boldsymbol{f}_{\mathrm{E}}$			
Layer Type	Weight Shape*	Output Shape	Activation
Fully Connected	300×784	300×1	ReLu
Fully Connected	64×512	64×1	ReLu
Fully Connected	37×128	37×1	ReLu
Fully Connected	10×64	10×1	ReLu
Decoder Network, $\boldsymbol{f}_{\mathrm{D}}$			
Layer Type	Weight Shape*	Output Shape	Activation
Fully Connected	49×10	49×1	ReLu
Fully Connected	93×64	93×1	ReLu
Fully Connected	350×128	350×1	ReLu
Fully Connected	784×512	784×1	Linear

* The initial output size in the encoder was 512, 128, 64, and 16, and the initial output size in the decoder was 64, 128, 512, and 784. The presented shapes are after removing dead nodes as described in Section 3.3.7.

3.2.2 Autoencoders with Artificial Neural Networks

An autoencoder consists of two functions. The first is an encoder function that takes some input $\boldsymbol{x} \in \mathbb{R}^{N\times 1}$, and encodes it into $\boldsymbol{z} \in \mathbb{R}^{M\times 1}$ where $M < N$. The second function acts as a decoder that takes the "code" $\boldsymbol{z}$ and reconstructs $\boldsymbol{x}$. Apart from compressing data, this is an effective way of removing noise from $\boldsymbol{x}$ as this will typically be lost in the encoding process. It is also a good way to measure how close two high dimensional samples are by instead measuring the distance in the encoded or latent space $\boldsymbol{z}$. It is for this purpose we use autoencoders in Chapter 6.

A simple way to realise the encoder and decoder functions, is with affine transforms i.e.

$$\boldsymbol{z} = \boldsymbol{f}_{\mathrm{E}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \tag{3.14}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{f}_{\mathrm{D}}(\boldsymbol{z}) = \boldsymbol{B}\boldsymbol{z} + \boldsymbol{d} \tag{3.15}$$

where $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ and $\boldsymbol{B} \in \mathbb{R}^{N\times M}$. The choice of $\boldsymbol{A}$ and $\boldsymbol{B}$ that minimises the reconstruction error $||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2$, is obtained with principal component analysis (PCA) as follows. Let $\boldsymbol{\Sigma} \in \mathbb{R}^{N\times N}$ be the covariance matrix of $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{\mu}$ where $\boldsymbol{\mu}$ is the mean value of $\boldsymbol{x}$. Assume that these are either known or that we have good estimates of these obtained e.g. via maximum likelihood from a number of samples of $\boldsymbol{x}$. Let $\boldsymbol{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_N] \in \mathbb{R}^{N\times N}$ be the eigenvector matrix of $\boldsymbol{R}$ with corresponding eigenvalues $\lambda_1, ..., \lambda_N$ in descending order. An encoder using PCA

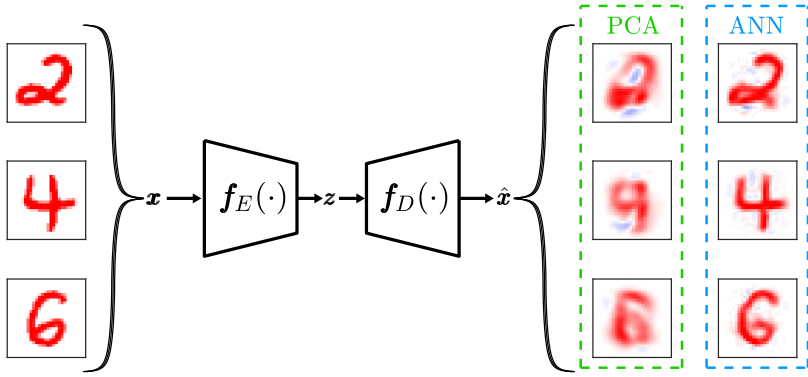


Figure 3.4: Example of an autoencoder using linear PCA and the non-linear ANN in Table 3.2 on MNIST digits. The size of the vector $\boldsymbol{z}$ is 10 in both cases.

is then obtained by selecting the M columns of $\boldsymbol{U}$ that correspond to the largest eigenvalues, forming $\boldsymbol{A}$ and $\boldsymbol{b}$ in (3.14) as

$$\boldsymbol{A} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_M]^T \tag{3.16}$$

$$\boldsymbol{b} = -\boldsymbol{A}\boldsymbol{\mu} \tag{3.17}$$

and the decoder in (3.15) is obtained as

$$\boldsymbol{B} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_M] \tag{3.18}$$

$$\boldsymbol{d} = \boldsymbol{\mu}. \tag{3.19}$$

A more detailed description of PCA is given in [61, Ch. 12].

A non-linear alternative is to use ANNs as the encoder and decoder functions. We will proceed with an example of this, again using the MNIST dataset of digit images. Consider the ANN based autoencoder in Table 3.2. This will take the 28×28 pixel input image, or 784 element vector, and encode it to a dimension of 10. The reconstruction error, measured as the mean squared error (MSE), on the training and test datasets were after 160 epochs of training 10.8 and 11.4, respectively, see Section 3.3. The corresponding losses using linear PCA was 27 and 26.7. Note that the "training" in PCA consists of estimating the mean and covariance matrix from the training data. Fig. 3.4 shows some examples of the reconstruction using the linear and nonlinear models. We can see a clear improvement using the non-linear method.

In Section 3.1.1, we discussed how a ReLu activated ANN can be regarded

as a data dependent affine transform in (3.6). The autoencoder in Table 3.2 acts as a different transform for all 70,000 MNIST samples. In other words, the input space with dimension 784 is partitioned into at least 70,000 linear regions with one sample in each.

3.3 Training Artificial Neural Networks

The training, optimisation, or learning of ANNs can be done in the form of supervised learning, unsupervised learning, and sometimes as semi-supervised learning which is a mixture of these. The distinction between what is supervised and what is unsupervised is however not always clear. In either case, some sort of prior direction of what is "good" and what is "bad" is necessary in order for learning to take place.

Supervised learning requires ground truth labels, or the true predictions that are associated with each input. In some cases it is possible to extract the label from the data directly, such as predicting time series or training an autoencoder as we saw in Section 3.2.2. In a series of data, such as words in a sentence, we can simply remove one word and train the ANN to predict the missing word. In an autoencoder, we train an ANN to recreate the input from an encoded version of the input, i.e. the input also acts as label. These scenarios are often referred to as unsupervised, or self-supervised, although the training process may be identical to a supervised learning setup. Another example of unsupervised learning is clustering which groups data, based on some given metric. An informal description of unsupervised learning is "most attempts to extract information from a distribution that do not require human labor to annotate examples" [70, Sec. 5.8]. In this thesis we make much use of data generated from models, and consequently, such data examples are automatically annotated. This would however in most cases still be described as supervised learning. The training of the classifier on the 77 GHz radar dataset in Chapter 6, is a typical example of supervised learning since the class labels of all 75,868 data samples were manually assigned.

3.3.1 Loss Functions

A good loss function is crucial for training ANNs. This scalar valued function describes the desired behavior of the ANN, or more specifically, it quantifies an error that we want to minimize. An example of a loss function that we will use is

the empirical MSE loss

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = L(\boldsymbol{y}_1, ..., \boldsymbol{y}_N, \hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_N) = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i||_2^2 \tag{3.20}$$

where $\boldsymbol{y}$ is the label and $\hat{\boldsymbol{y}}$ is the predicted output from the ANN. Here we have simplified the notations such that $\boldsymbol{y}$ represents all N samples $\boldsymbol{y}_1, ..., \boldsymbol{y}_N$ and likewise for $\hat{\boldsymbol{y}}$. Another common loss function we will use is the categorical cross-entropy loss. This is suitable for training classifiers with one hot encoding as labels. Let D be the number of classes and let the label be $\boldsymbol{y} \in \mathbb{R}^{D \times 1}$, which is zero valued at all elements except one, indicating the true class. The categorical cross-entropy loss is then

$$\begin{aligned} L(\boldsymbol{y}, \hat{\boldsymbol{y}}) &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \boldsymbol{y}_i[d] \log(\hat{\boldsymbol{y}}_i[d]) \\ &= -\frac{1}{N} \sum_{i=1}^{N} \log(\boldsymbol{y}_i^T \hat{\boldsymbol{y}}_i). \end{aligned} \tag{3.21}$$

If we treat $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ as discrete probability vectors (which they in general are not), then minimizing (3.21) is equivalent to minimising the empirical Kullback–Leibler divergence between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ given as

$$\begin{aligned} D_{\mathrm{KL}}(\boldsymbol{y}||\hat{\boldsymbol{y}}) =& \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \boldsymbol{y}_i[d] \log \left(\frac{\boldsymbol{y}_i[d]}{\hat{\boldsymbol{y}}_i[d]} \right) \\ =& \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \boldsymbol{y}_i[d] \log \left(\boldsymbol{y}_i[d] \right) - \boldsymbol{y}_i[d] \log \left(\hat{\boldsymbol{y}}_i[d] \right) \qquad (3.22) \\ =& P(\boldsymbol{y}) - \frac{1}{N} \sum_{i=1}^{N} \sum_{d=1}^{D} \boldsymbol{y}_i[d] \log \left(\hat{\boldsymbol{y}}_i[d] \right). \qquad (3.23) \end{aligned}$$

The equivalence assumes that the term $P(\boldsymbol{y})$ is constant which, for a given dataset, it is. The Kullback–Leibler divergence can be regarded as a distance between two distributions, hence the motivation for using it as a loss function in form of the categorical cross-entropy loss in (3.21). To obtain normalized scores in $\hat{\boldsymbol{y}}$, the softmax output is often used and is given as

$$\hat{\boldsymbol{y}}[d] = \frac{e^{\boldsymbol{z}[d]}}{\sum_{i=1}^{D} e^{\boldsymbol{z}[i]}} \tag{3.24}$$

for $d = 1, ..., D$, where $\boldsymbol{z}$ is the linear output from the final layer. Another reason for using the softmax normalization with one hot encoded labels and the categorical cross entropy loss, is that the gradient for each sample

$$\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}} = \frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}} \tag{3.25}$$

where

$$\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} = \frac{\partial}{\partial \hat{\boldsymbol{y}}} - \log(\boldsymbol{y}^T \hat{\boldsymbol{y}}) = -\frac{\boldsymbol{y}}{\boldsymbol{y}^T \hat{\boldsymbol{y}}} \tag{3.26}$$

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}} = \frac{\partial}{\partial \boldsymbol{z}} \frac{e^{\boldsymbol{z}}}{\mathbf{1}^T e^{\boldsymbol{z}}} = \text{diag}(\hat{\boldsymbol{y}}) - \hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^T \tag{3.27}$$

simplifies to

$$\begin{aligned} \frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{z}} &= -\frac{\boldsymbol{y}^T}{\boldsymbol{y}^T \hat{\boldsymbol{y}}} \left[\text{diag}(\hat{\boldsymbol{y}}) - \hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^T \right] \\ &= -\frac{(\boldsymbol{y} \circ \hat{\boldsymbol{y}})^T}{\boldsymbol{y}^T \hat{\boldsymbol{y}}} + \frac{\boldsymbol{y}^T \hat{\boldsymbol{y}} \hat{\boldsymbol{y}}^T}{\boldsymbol{y}^T \hat{\boldsymbol{y}}} \\ &= -\boldsymbol{y}^T + \hat{\boldsymbol{y}}^T = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T. \end{aligned} \tag{3.28}$$

The last step comes from noting that $\boldsymbol{y}$ is a one hot vector. Here $\mathbf{1}$ in (3.27) is a vector of ones and $\text{diag}(\boldsymbol{x})$ in (3.27) is a diagonal matrix with the values of $\boldsymbol{x}$ along the diagonal. Calculating gradients is an important step in backpropagation used when optimising ANNs, where this calculation will be repeated many times. Any simplification that can be made, such as the one in (3.28), may thus save both time and energy.

3.3.2 Backpropagation

ANNs are optimised to minimise the loss function via an application of the chain rule and gradient descent, known as backpropagation. When the ANN consists of multiple layers it is considered to be a deep neural network, and consequently the training/learning process has come to be known as deep learning. In backpropagation, we aim to calculate the gradient of the loss function with respect to all the parameters in the network. We then update the parameters according to some gradient descent scheme until the loss no longer decreases. As we will see, there are factors that will complicate this process. We begin with calculating the gradient.

Let the set of parameters in layer k be $\boldsymbol{\theta}_k$, e.g. in a fully connected layer we have $\boldsymbol{\theta}_k = \{\boldsymbol{W}_k, \boldsymbol{b}_k\}$. Let the output from layer k be $\boldsymbol{y}_k = a_k(\boldsymbol{z}_k) \in \mathbb{R}^{D_k \times 1}$ where $a_k(\cdot)$

is the activation function and $\boldsymbol{z}_k \in \mathbb{R}^{D_k \times 1}$ is the input to the activation function, e.g. $\boldsymbol{z}_k = \boldsymbol{W}_k \boldsymbol{y}_{k-1} + \boldsymbol{b}_k$. Note that in the final layer K, $\boldsymbol{y}_K = \hat{\boldsymbol{y}}$. From the chain rule we have that

$$\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{\theta}_k} = \frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{y}_k} \frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{z}_k} \frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{\theta}_k} \tag{3.29}$$

where

$$\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{z}_k} = \frac{\partial}{\partial \boldsymbol{z}_k} a_k(\boldsymbol{z}_k) \in \mathbb{R}^{D_k \times 1} \tag{3.30}$$

and

$$\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{y}_k} = \frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{y}_{k+1}} \frac{\partial \boldsymbol{y}_{k+1}}{\partial \boldsymbol{z}_{k+1}} \frac{\partial \boldsymbol{z}_{k+1}}{\partial \boldsymbol{y}_k} \in \mathbb{R}^{D_k \times 1}. \tag{3.31}$$

The derivative in (3.29) should be interpreted as the partial derivative of the loss with respect to all the parameter values in $\boldsymbol{\theta}_k$, having the same structure as $\boldsymbol{\theta}_k$. The gradient in (3.31) in the second to last layer can be written as

$$\begin{aligned} \frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{y}_{K-1}} &= \frac{\partial L(\boldsymbol{y}, \boldsymbol{y}_K)}{\partial \boldsymbol{y}_{K-1}} \\ &= \frac{\partial L(\boldsymbol{y}, \boldsymbol{y}_K)}{\partial \boldsymbol{y}_K} \frac{\partial \boldsymbol{y}_K}{\partial \boldsymbol{z}_K} \frac{\partial \boldsymbol{z}_K}{\partial \boldsymbol{y}_{K-1}}. \end{aligned} \tag{3.32}$$

We thus see that starting at the last layer, we can calculate (3.32) and recursively use the results to calculate (3.31) for $k = K-2, ..., 1$ propagating backwards through the network. At each layer we then use (3.31) to calculate (3.29).

Continuing with our fully connected example where $\boldsymbol{\theta}_k = \{\boldsymbol{W}_k, \boldsymbol{b}_k\}$, let

$$\boldsymbol{g}_k = \frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{y}_k} \frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{z}_k} \in \mathbb{R}^{D_k \times 1} \tag{3.33}$$

which we have obtained iteratively using (3.31) and (3.30) starting at $k = K$. We then obtain (3.29) as

$$\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{W}_k} = \boldsymbol{g}_k \boldsymbol{y}_{k-1}^T \in \mathbb{R}^{D_k \times D_{k-1}} \tag{3.34}$$

$$\frac{\partial L(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \boldsymbol{b}_k} = \boldsymbol{g}_k \in \mathbb{R}^{D_k \times 1} \tag{3.35}$$

If $a_k(\cdot)$ is the ReLu function, we calculate (3.30) as

$$\frac{\partial}{\partial \boldsymbol{z}_k} a_k(\boldsymbol{z}_k[i]) = a_k'(\boldsymbol{z}_k[i]) = \begin{cases} 1 & \text{if } \boldsymbol{z}_k[i] > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.36}$$

To do these calculations we will need values for $\boldsymbol{z}_k$ for $k = 1, ..., K$ and for $\boldsymbol{y}_k$ for $k = 0, ..., K$ where $\boldsymbol{y}_0 = \boldsymbol{x}$ and $\boldsymbol{x}$ is the network input. These are obtained by first evaluating the entire network in a forward propagation. We then calculate the gradients in the backward propagation and update the parameter values. This process is repeated in each iteration. Doing these calculations mostly involve matrix multiplications and can therefore be efficiently calculated using parallelization in hardware designed for such computations, such as graphics processing units (GPUs).

The first published work on using a GPU for training a CNN was in 2006 [71], although this implementation was not as flexible as later versions such as [72] in 2011, and the training of AlexNet in 2012 [73]. This development and implementation in hardware was a significant milestone in the development of ANNs, and consequently for AI in general, as it significantly reduced the training time. The reduction in training time cannot be stated in general as it depends on network architecture and the optimisation setup. As an example, a speedup factor of 60 was reported in [72] which is quite significant considering that the training still took up to 10 hours on a GPU. The ANNs in this thesis were trained using Keras [74] with Tensorflow [75] as backend on an Nvidia Geforce RTX 2080 GPU.

We have for the most part used ReLu activations in this thesis, the exceptions being the final activations which should be matched to the label. The reason for using ReLu activations is twofold. First it is cheap to evaluate both the activation and the gradient in (3.36). Second, it does not suffer from the vanishing gradient problem, at least not to the same extent as e.g., sigmoid activated networks. The sigmoid activation is given as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.37}$$

and is bounded to the interval [0, 1]. It is therefore a natural extension to the original threshold based activation used in early ANNs, but has the advantage of being differentiable. This derivative is however at most 0.25 (when $z = 0$). Multiplying the gradient by a number < 0.25 in each layer during backpropagation may cause the gradient to become "vanishingly" small as it backpropagates deeper into the network. This can significantly slow down, or even stop, the learning process.

3.3.3 Stochastic Gradient Descent

A dataset with N samples will give us N different gradients, from which we should take the average when performing the gradient descent steps. However, doing the forward and backward passes for all N samples in every iteration may result in a heavy computational load since N typically needs to be quite large in order for the ANN to generalise well. This is overcome by instead calculating the gradients for a smaller batch with N_b samples, and only averaging over these in each iteration. A different batch is chosen in the next iteration until the whole dataset has been used. This is referred to as an epoch, and the number of iterations in an epoch is thus $\lceil N/N_b \rceil$ where $\lceil \cdot \rceil$ denotes the ceiling function. An ANN is typically trained over a number of epochs. Since the gradients will be different for different batches/iterations, the gradient descent process becomes noisy and is known as stochastic gradient descent. We use stochastic gradient descent in the training of all ANNs in this thesis.

3.3.4 Adaptive Moment Estimation

An effective way to reduce the randomness in stochastic gradient descent, is the use of momentum where the current gradient is averaged with previous gradients using an exponential decay. This may also help in reaching a better local optimum, as the optimisation problem of training an ANN is typically non-convex. Provided a sufficiently small step size, non-convex gradient descent optimisation is guaranteed to reach a local optimum, or at lest the proximity of a local optimum if using stochastic gradient descent. By introducing momentum, it becomes possible to continue in directions that have on average been descending when these locally shift to an ascension. This is illustrated in Fig. 3.5 where we have used the adaptive moment estimation (Adam) optimisation method [76].

Adam combines momentum with a version of root mean square propagation (RMSProp), a method introduced by Geoffrey Hinton in 2012. In RMSProp the step size is adapted to each parameter by dividing it by a filtered root mean square (RMS) value of the gradient. This yields a smaller step size for parameters with large gradients and larger step size for parameters with small gradients, leading to a more effective convergence in general. Adam optimisation is given as follows. A parameter θ at iteration i is updated as

$$\theta^{(i)} = \theta^{(i-1)} - \eta \frac{\hat{m}_\theta^{(i)}}{\sqrt{\hat{v}_\theta^{(i)}} + \epsilon} \tag{3.38}$$

where η is a stepsize, ϵ is a small number that prevents division by zero, $\hat{m}_\theta^{(i)}$ and

Table 3.3: Default Parameter Values for Adam

Parameter	Value	Description
η	0.001	Stepsize or learning rate
ϵ	10^{-8}	Denominator bias, prevents division by zero.
β_1	0.9	Exponential decay for the first moment of the gradient. Used for momentum.
β_2	0.999	Exponential decay for the second moment of the gradient. Used for "RMSProp".

$\hat{v}_\theta^{(i)}$ are the momentum and RMSProp variables respectively for the parameter θ. These are obtained as

$$\hat{m}_\theta^{(i)} = \frac{m_\theta^{(i)}}{1-\beta_1^i} \tag{3.39}$$

$$\hat{v}_\theta^{(i)} = \frac{v_\theta^{(i)}}{1-\beta_2^i} \tag{3.40}$$

where

$$\begin{aligned} m_\theta^{(i)} &= \beta_1 m_\theta^{(i-1)} + (1-\beta_1)\bar{g}^{(i)} \\ v_\theta^{(i)} &= \beta_2 v_\theta^{(i-1)} + (1-\beta_2)\left(\bar{g}^{(i)}\right)^2 \end{aligned} \tag{3.41}$$

$$\bar{g}^{(i)} = \frac{\partial \bar{L}(\boldsymbol{y}, \hat{\boldsymbol{y}})}{\partial \theta}^{(i)} \tag{3.42}$$

and β_1 and β_2 are parameters and $\bar{g}^{(i)}$ is the mean gradient from a batch with respect to θ at iteration i. Note that the notation is such that $x^{(i)}$ is the value of x at iteration i while x^i is x to the power of i. In this thesis we use Adam optimisation in the training of all ANNs with the default parameter values in Table 3.3. The use of the squared gradient via $\hat{v}_\theta^{(i)}$ in (3.38), is identical to RMSProp apart from the bias correction in (3.40).

Our intuition from optimisation in low dimensional space might tell us that gradient descent in non-convex problems will lead to poor solutions i.e., there may be one good global optimum with many poor local optima. This turns out not to be as problematic with high dimensional ANNs, i.e. high dimensional in both the input space as well as the parameter space. We will not go into any discussion as to why this is. We simply note that in practice, training ANNs with stochastic gradient descent works quite well, especially when using optimisation schemes such as Adam. It is true that different parameter initialisation will lead to different solutions. However, such differences observed in the studies presented in

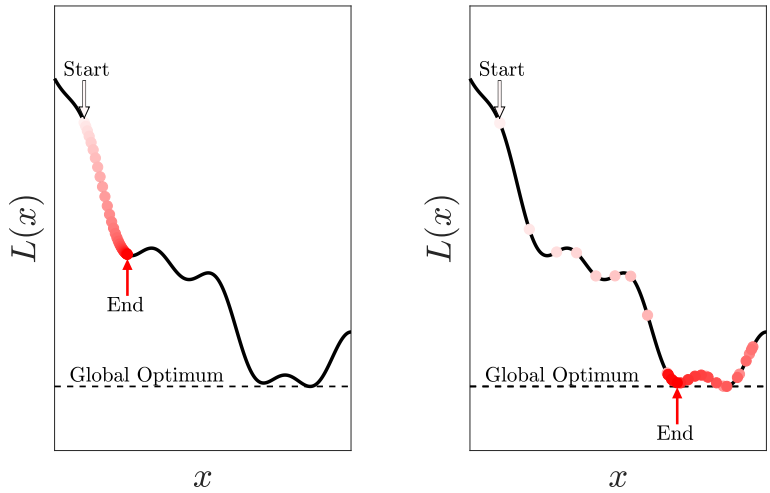


Figure 3.5: Examples of gradient descent using a fixed step size (a) and Adam (b). The plots show the start point and the final end point where the loss has converged. The number of iterations was 3000. Every 100'th iteration is marked ranging from light pink color (early iterations) to red color (final iterations). Due to the adaptive stepsize in Adam, the solution converges much faster to the region of the global optimum. Due to momentum it does not get stuck in local optima on the way, unlike the regular gradient descent in (a).

this thesis have not been very significant. Training the ANNs twice with different initialisations has almost always lead to more or less the same loss level. This is however not true in general.

3.3.5 Parameter Initialisation

All optimisation methods require some initial values of the parameters that are to be optimised. For this we have used Xavier initialisation [77] and He [78] initialisation. These initialisations are designed for sigmoid activated networks and ReLu activated networks respectively. In Xavier (or Glorot uniform) initialisation, the weights are drawn from a uniform distribution with limits $\pm\sqrt{6/(D_k + D_{k-1})}$ where D_k is the number of output nodes and D_{k-1} is the number of input nodes in layer k. In He initialisation, the parameters are drawn from a Gaussian distribution with zero mean and standard deviation $\sqrt{2/D_{k-1}}$.

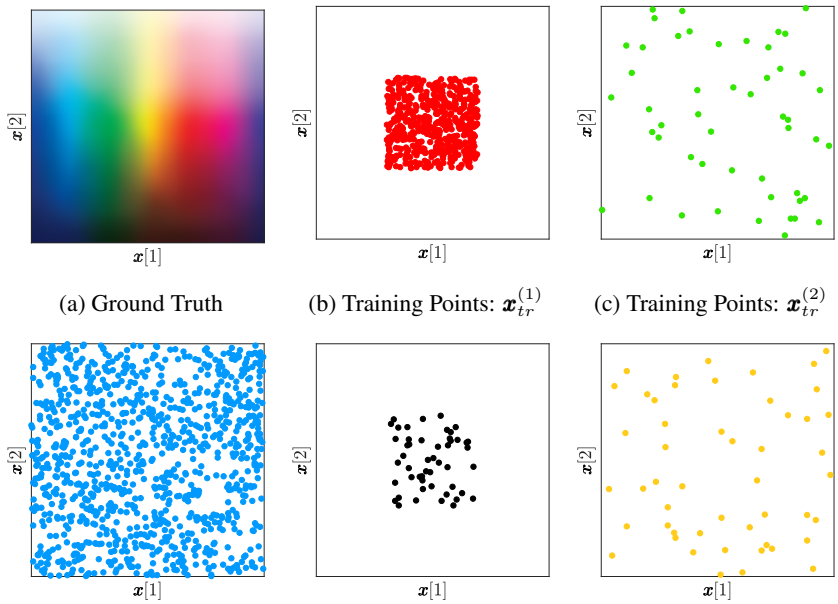
3.3.6 Training, Validation and Testing Data

We mentioned in Section 3.1.1 that ANNs can be regarded as universal function approximators. Even if we assume that this is true for a particular ANN, we will still be limited by the diversity of the data. It is after all the sampled data that will determine the final function of the ANN, and not the function that the data is sampled from. This may result in an ANN that performs well on the data it is trained on, but is in general not a good match to the desired function. In the field of machine learning this is known as overfitting and is typically evaluated by partitioning the data into two or three different sets for training, validation and testing.

The training data is what is used in the optimisation procedure, typically backpropagation in the case of ANNs. The validation data is used only for evaluation. Overfitting is then indicated when the loss from evaluating the ANN on the training data differs from the loss obtained with the validation data. The greater the difference, the greater the indication of overfitting. In such situations one could implement various regularisation methods for the purpose of reducing this difference. There are many approaches for regularisation and it typically involves the tuning of some parameter values, referred to as hyperparameters. These are not explicitly optimised by any algorithm. Instead the operator tunes these until a satisfactory functional behavior is reached. This manual optimisation may also result in overfitting to both the training and validation data. To asses this, it is common to also have a third dataset for the final testing, i.e. a test dataset. In some cases it is sufficient with a single dataset for both validation and testing.

Our studies in Chapter 6 concern the importance of selecting appropriate data when training ANNs. As a background to this, we give a simple example. Assume that we want to find a function that maps $\boldsymbol{x} \in \mathbb{R}^{2\times 1}$ to $\boldsymbol{y} \in \mathbb{R}^{3\times 1}$ using an ANN. In this case $\boldsymbol{x}$ is a Cartesian coordinate and $\boldsymbol{y}$ is a RGB color coordinate. The whole desired function of interest is shown in Fig. 3.6a. We now consider three cases where we sample this space according to the three scenarios in Fig. 3.6b-3.6d. In the first scenario in Fig. 3.6b, the data is sampled quite densely but only within a limited area. In the second scenario in Fig. 3.6c, the full space is sparsely sampled and in the third scenario in Fig. 3.6d the full space is densely sampled. We also sample two datasets for testing as shown in Fig. 3.6e and Fig. 3.6f. For each scenario, we train the MLP in Table 3.4 using stochastic gradient descent and Adam optimisation with the MSE loss in (3.20). In this particular case we reduced the learning rate in Table 3.3 to $\eta = 0.0001$ since it improved the learning in all scenarios.

The results for scenario 1 are shown in Fig. 3.7a. The MLP behaves quite well within the training dataset (green-yellow-red region) and performs poorly outside this area. This is also indicated by the significant difference between the two test losses for $\boldsymbol{x}_{te}^{(1)}$ and $\boldsymbol{x}_{te}^{(2)}$. We should not be surprised by this results since we have



(a) Ground Truth (b) Training Points: $\boldsymbol{x}_{tr}^{(1)}$ (c) Training Points: $\boldsymbol{x}_{tr}^{(2)}$

(d) Training Points: $\boldsymbol{x}_{tr}^{(3)}$ (e) Test Points: $\boldsymbol{x}_{te}^{(1)}$ (f) Test Points: $\boldsymbol{x}_{te}^{(2)}$

Figure 3.6: Examples of different training and testing sets. The aim is to approximate the function in (a). In (b) we show scenario 1 with 500 training samples. In (c) we show scenario 2 with only 50 training samples. In (d) we see scenario 3 where we have 1000 training samples. We have two test sets shown in (e) and (f) with 50 samples in each.

given no means for the MLP to learn the functional behaviour outside the sampled region.

The results for scenario 2 are shown in Fig. 3.7b. This performs very well on the training data in terms of the loss, but generalises poorly and has high losses for both test sets. This poor performance is however different from the results in scenario 1. Here we have some reason to believe that the MLP could learn the functional behavior by interpolating between samples. Our problem in this case is that the chosen architecture yields a too versatile function. By imposing regularisation constraints we can reduce this and thereby gain a solution that is more consistent between training data and the test data. One method for doing this is dropout [79], where a node in a particular layer is randomly set to zero with some probability p in each iteration. Once trained the dropout is no longer used. The result can be regarded as an average of several smaller networks where the averaging reduces

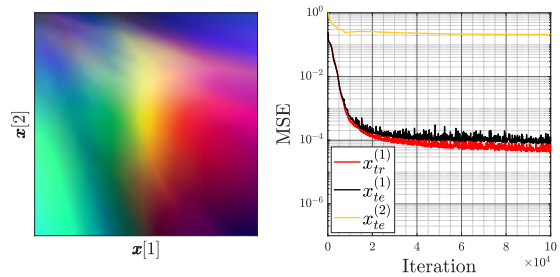
Table 3.4: MLP for RGB Colors

Layer Type	Weight Shape	Output Shape	Activation
Fully Connected	64×2	64×1	ReLu
Fully Connected	128×64	128×1	ReLu
Fully Connected	64×128	64×1	ReLu
Fully Connected	32×64	32×1	ReLu
Fully Connected	16×32	16×1	ReLu
Fully Connected	3×16	3×1	Linear

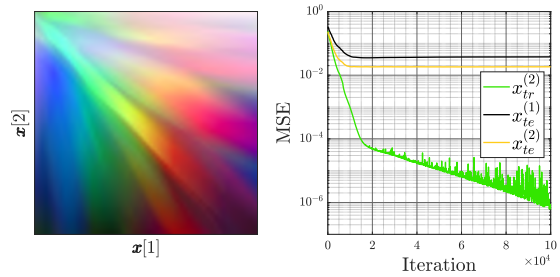
Total number of parameters: 19,427, including bias parameters.

the expressivity of the ANN. We show an example of this in Fig. 3.7c where we have added dropout to the second and fourth layer with dropout rate $p = 0.1$ in both layers. These hyperparameters were more or less arbitrarily chosen and it is possible that different values would yield better results. It is however clear that the testing and training losses are now more consistent and the regularisation has thus had the desired effect.

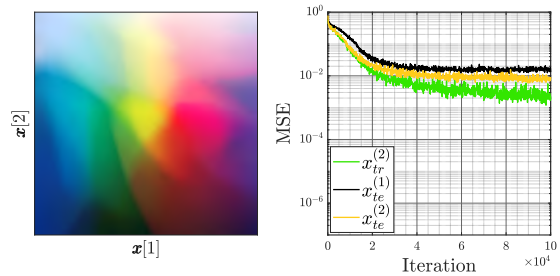
Although the results in Fig. 3.7c are an improvement from those in Fig. 3.7b, we still note some considerable differences between the color chart in Fig. 3.7c and the ground truth in Fig. 3.6a. Correcting this can only be done by introducing more information about the desired function, such as more regularisation, smoothing etc. This can also be achieved ignorantly via more or less lucky guesses when experimenting with the design of the ANN. Alternatively we can add more data samples, as we have done in scenario 3. The results for this are shown in Fig. 3.7d. Here the color chart appears to be quite close to the ground truth in Fig. 3.6a, with only some slight artifacts. This improved generalisation is also indicated by the training and test losses in Fig. 3.7d, which are on a more consistent and lower level than those in Fig. 3.7a-3.7c. In this thesis we will mostly operate in this densely sampled regime since we sample data from models and there is no limit to how much data we can sample. This also removes the need for having an extra test dataset in addition to the validation dataset.



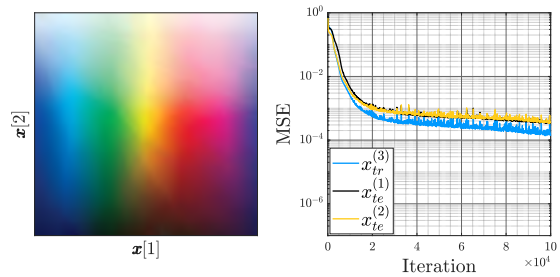
(a) Trained on $\boldsymbol{x}_{tr}^{(1)}$.



(b) Trained on $\boldsymbol{x}_{tr}^{(2)}$.



(c) Trained on $\boldsymbol{x}_{tr}^{(2)}$ with dropout.



(d) Trained on $\boldsymbol{x}_{tr}^{(3)}$.

Figure 3.7: Training and test results for the three scenarios in Fig. 3.6.

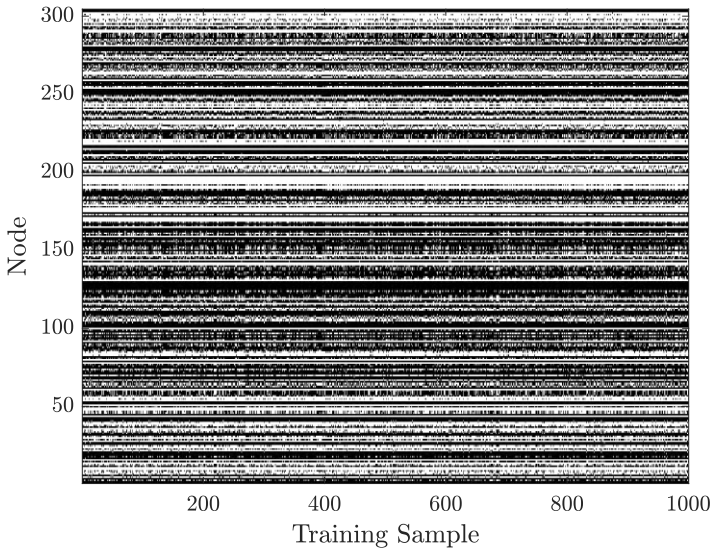


Figure 3.8: Activations of the MLP in Table 3.4 for all 1000 training samples. The network had 304 nodes with ReLu activations. A white color indicates that the node is active, i.e. the input to the ReLu function is positive, and black color indicates that the node is inactive, i.e. the input to the ReLu function is non-positive. Several rows/nodes are inactive for all training samples.

3.3.7 Network Pruning

An ANN will typically have many parameters and it is often the case that some of these can be removed without changing the function. For ReLu activated networks this can be done quite easily by removing nodes that are never active in any training sample. An example is shown in Fig. 3.8, where the activations for the MLP in Table 3.4 are shown for scenario three, cf. Fig. 3.6d and Fig. 3.7d. In this particular case 59 out of the 304 nodes with ReLu activation were inactive for all training samples and could thus be removed. The total number of parameters was then reduced by almost 20% from 19,427 to 15,760. We have used this type of network pruning in the final optimisation of many of the ANNs presented in this thesis.



Figure 3.9: "La trahison des images" (The Treachery of Images) by René Magritte, 1929. © 2023 Digital Image Museum Associates/LACMA/Art Resource NY/Scala, Florence.

3.3.8 Understanding the Function

René Magritte's painting "La trahison des images", shown in Fig. 3.9, depicts a pipe with the text "Ceci n'est pas une pipe" (This is not a pipe). Pointing out the difference between an actual pipe and an image of a pipe may be trivial, but it is an important distinction to make when discussing ANNs. In the previously discussed digit classification problem in Section 3.2.1, we were not training the ANN to learn the difference between different numbers. We simply trained the ANN to distinguish between samples from different distributions. Changing a distribution, e.g. by rotating an image, will be equivalent to sampling data outside the distribution of the training data, resulting in situations illustrated with scenario 1 in Fig. 3.6 and Fig. 3.7a. Similar effects can be obtained by arbitrarily scaling the data sample. For this reason it is common to first normalize the input data before passing it to an ANN.

It cannot always be assumed that an ANN with near perfect classification accuracy has learnt the desired function. Or phrased differently, it cannot always be assumed that the training data accurately represents the desired function. This is particularly true when using real sensor data since there may be artifacts that

can be used as unintended distinguishing features. This is discussed in [80], where the authors give an example of an ANN classifier trained for classifying images of either a wolf or a husky dog. These were such that all images of wolves had snow in the background and consequently an image was classified as wolf if the background had snow or was light, regardless of the animal in the picture. In signal processing, these type of artifacts may appear as distortions in the signal caused by the sensor, or simply by different signal classes having different SNR distributions. Such "artifacts" are easier to manage when training on synthetic data, since we then have full insight of what the data contains through the model.

Having this in mind, it may be desirable to evaluate why a sample was classified to a particular class by an ANN classifier. This is not always simple to do, and ANNs are for reasons like this often described as "black boxes". One method that we will look into in Chapter 6 is to examine the data dependent equivalent filters that a ReLu activated ANN represents for a given sample. In such cases we can express the ANN as

$$\hat{\boldsymbol{y}} = \tilde{\boldsymbol{W}}\boldsymbol{x} + \tilde{\boldsymbol{b}}. \tag{3.43}$$

where the class is assigned as the argument of the maxima of $\hat{\boldsymbol{y}} \in \mathbb{R}^{D_K \times 1}$, where $\boldsymbol{x} \in \mathbb{R}^{D_0 \times 1}$ is the input and $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{b}}$ can be obtained iteratively as

$$\tilde{\boldsymbol{W}}_k = (\boldsymbol{W}_k \circ \boldsymbol{A}_k)\,\tilde{\boldsymbol{W}}_{k-1} \tag{3.44}$$

$$\tilde{\boldsymbol{b}}_k = \tilde{\boldsymbol{W}}_k \tilde{\boldsymbol{b}}_{k-1} + \boldsymbol{b}_k \circ a_k'(\boldsymbol{z_k}) \tag{3.45}$$

$$\boldsymbol{z}_k = \boldsymbol{W}_k \boldsymbol{y}_{k-1} + \boldsymbol{b}_k \tag{3.46}$$

$$\boldsymbol{y}_k = a_k(\boldsymbol{z}_k) \tag{3.47}$$

for $k = 1, ..., K$ where

- $\tilde{\boldsymbol{W}} = \tilde{\boldsymbol{W}}_K$ and $\tilde{\boldsymbol{b}} = \tilde{\boldsymbol{b}}_K$
- $\boldsymbol{W}_k \in \mathbb{R}^{D_k \times D_{k-1}}$ and $\boldsymbol{b} \in \mathbb{R}^{D_k \times 1}$ is the weight matrix and bias vector of layer k (for convolutional layers these are the equivalent convolutional matrix and bias vector)
- $\circ$ denotes the Hadamard product
- $\boldsymbol{A}_k \in \mathbb{R}^{D_k \times D_{k-1}}$ is a binary valued matrix with values 0 and 1. The columns are repetitions of the vector $a_k'(\boldsymbol{x}) \in \mathbb{R}^{D_k \times 1}$ which is the gradient vector in (3.36)
- we initialise with $\tilde{\boldsymbol{W}}_0 = \boldsymbol{I} \in \mathbb{R}^{D_0 \times D_0}$ where $\boldsymbol{I}$ is the identity matrix, $\tilde{\boldsymbol{b}}_0$ is zero valued and $\boldsymbol{y}_0 = \boldsymbol{x}$.

We saw this description of a ReLu activated ANN earlier in (3.6) in Section 3.1.1.

Let $\tilde{\boldsymbol{w}}_d$ be the d'th row of $\tilde{\boldsymbol{W}}$, and $\tilde{b}_d$ the d'th element of $\tilde{\boldsymbol{b}}$. A high score, or the value of $\tilde{\boldsymbol{w}}_d\boldsymbol{x} + \tilde{\boldsymbol{b}}_d$, is an indication of a good match of $\boldsymbol{x}$ to class d. By examining what part of $\boldsymbol{x}$ is weighted by $\tilde{\boldsymbol{W}}$, we may obtain some insight to what features in $\boldsymbol{x}$ the classification is based on. Note that this will not always give intuitive results. We illustrate this with an example.

Consider again the problem of classifying digits presented in Section 3.2.1. We use the same CNN in Table 3.1 which does not have any bias parameters to better illustrate the concept of equivalent filters. The CNN was trained for 50 epochs with stochastic gradient descent and Adam optimisation. The losses then converged, yielding a final training and validation accuracy of 98.2% and 97.7%, respectively. The accuracy is simply calculated as the number of correctly classified samples divided by the total number of samples in each dataset. The equivalent filters for all ten classes are shown in Fig. 3.10 for two examples. At first glance these may not offer much intuition. There are however some patterns that are indeed intuitive.

The contour of the digits is outlined in all figures (black curves). The input is zero valued outside this contour. The weight values outside the contour area in the filters are therefore irrelevant to the score value. As expected, a lot of positive weight is put within this contour for the correct filters of 0 and 7. In Fig. 3.10a, we see that filer 0 and 6 are similar except filter 6 has negative weight where one would expect an opening in the circle for a 6 digit. In Fig. 3.10b we see that a lot of weight is put in the horizontal line of the digit in filter 7, whereas this area has negative weight in filter 1. This is intuitive considering that it is this line that is the main distinguishing feature between a “1” and a “7”.

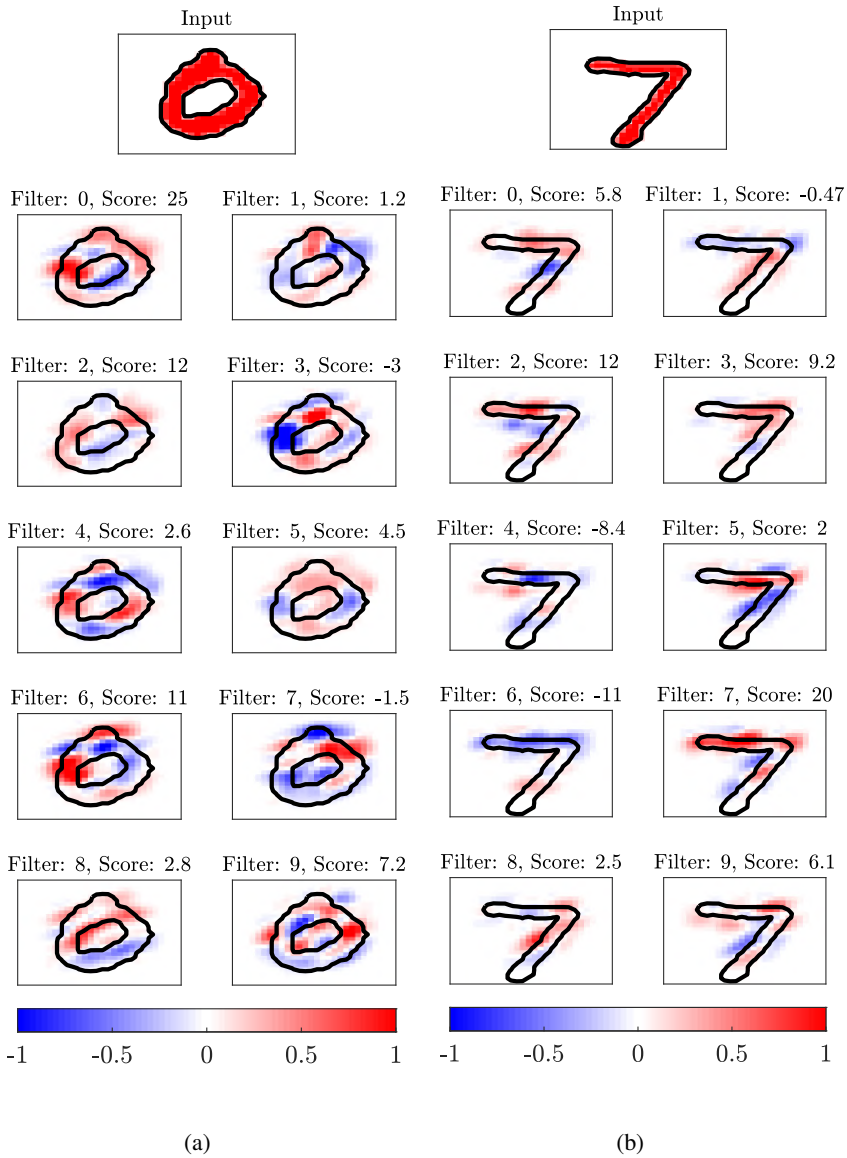


Figure 3.10: Examples of equivalent filters from the CNN in Table 3.1, for a "0" digit in (a) and a "7"' digit in (b), obtained from the MNIST dataset. The scores are before the softmax normalization and are the scalar product of the input and the respective filters.

Chapter 4

Stepped Frequency Pulse Compression with Non-Coherent Radar

In this chapter, a CNN is used to achieve subpulse resolution in non-coherent stepped frequency waveform radar. The results and methods presented in this chapter are published in paper A; see page xi.

4.1 Introduction

A key design parameter of any radar system is the range resolution which, as we saw in Section 2.2, has a theoretical limit [81]

$$\partial r = \frac{c}{2B}. \tag{4.1}$$

We saw in Section 2.3 that the bandwidth can be increased by various pulse compression techniques. As this often requires phase information the term "pulse compression" is almost synonymous with coherent radar.

There are however methods for non-coherent pulse compression, first introduced in [43] and further developed in [82], [83], [84], where the pulse is divided into a series of sub-pulses acting as on/off bits, referred to as on-off keying (OOK). When cross-correlated with a reference sequence a single peak/pulse is obtained with increased SNR. The resolution is determined by the length of each sub-pulse and the gain in SNR depends on the number of sub-pulses, i.e. having only one

sub pulse is equivalent to a standard pulsed radar. According to [85] such non-coherent pulse compression performs slightly worse in terms of SNR gain than simply integrating several consecutive pulses, but with the advantage of getting the gain within one PRI.

Here we present an alternative method for non-coherent pulse compression with the main purpose of increasing the range resolution without increasing the instantaneous bandwidth, i.e. maintaining a relatively long, uninterrupted pulse length. In other words, we consider radar systems with constraints on instantaneous bandwidth and with no phase information. As for constraints on instantaneous bandwidth, a common waveform for coherent systems is the stepped frequency waveform. This utilizes frequency agility where pulses are transmitted at different frequencies, yielding an increased total bandwidth over a set of pulses, and thereby higher range resolution. In this study we use a non-coherent stepped frequency waveform and a CNN acting as a pulse compression "filter."

Unlike the SNR gain, a resolution gain can be of significant value even at high SNRs. As an example, an SNR of 18 dB yields a detection probability of 99.99% and false alarm probability of 10^{-10} for a Rayleigh distributed target in AWGN [86, p. 17]. Increasing the SNR beyond this will thus only yield marginal improvements in the detection probability and false alarm trade-off. The resolution gain is however valuable regardless of the SNR. Note that for coherent systems the term "pulse compression" implies both these gains.

The method we present may be particularly suitable for non-coherent radar systems that require frequency agility for other reasons, such as minimizing interference, as it in such cases may only require additional signal processing rather than additional physical requirements on the radar system. It is also applicable to coherent systems that may lose coherence when shifting frequency or, as we will see, in complicated target situations. Non-coherent systems will always be inferior to their coherent counterparts in terms of performance. The reason for choosing a non-coherent system in the first place is instead based on other factors such as production cost, power criteria, and design complexity.

With the stepped frequency waveform and a coherent radar system, a high resolution range profile is achieved by taking the inverse discrete Fourier transform (IDFT) of a set of M pulses [87]. See Section 4.2.2. In a non-coherent case, we only have the magnitude of each pulse sample, resulting in an inverse phase problem where we aim to transform a complex vector from frequency to time/range domain without knowing the phase, $|\boldsymbol{G}(f)|^2 \rightarrow \boldsymbol{g}(t)$. As is the case here, this problem often arises with non-coherent radiation, i.e. unknown phase, such as X-ray and optics and has been widely studied; see e.g. [88]. Iterative algorithms exist for going from $|\boldsymbol{G}(f)|^2$ to $\boldsymbol{g}(t)$, [89], and requires some prior knowledge of $\boldsymbol{g}(t)$, e.g. that it is non-negative and real, or that the signal is sparse. Without prior knowledge of the time

domain signal the problem is impossible to solve since any random guess of the phase is then a valid solution. In our case the target is complex-valued, see Section 4.2.2, making the iterative solutions difficult to apply. Moreover, such iterative methods may be sensitive to initial values, and require an undetermined number of iterations.

Non-iterative and non-CNN based solutions for sparse phase retrieval have also been developed where K scattering points are resolved from the magnitude of the frequency spectrum [90]. However, the required number of samples in frequency domain is on the order K^2, which in this context may be very large.

Since we are mainly interested in retrieving the magnitude of $\boldsymbol{g}(t)$, we will instead set up a non-linear regression problem where we train a neural network to directly transform a frequency domain matrix $\boldsymbol{X}$ to a discrete estimate of the magnitude in the time domain, $\boldsymbol{y}$. We thereby never explicitly retrieve the phase. Since the execution time of a trained neural network is deterministic, this solution may be preferable over the aforementioned iterative solutions, an aspect that has been further studied in [91].

Although the use of deep learning has been studied in the field of phase retrieval, e.g. [91], [92], [93], and [94], it has to our knowledge not been applied in this context of non-coherent radar, and unlike most other such studies, we use an ANN as a direct magnitude-to-magnitude transformation, and do not recover the phase.

We generate data from a model to train the network on. Our results are therefore constrained to the validity of this model. The rest of this chapter is organized as follows. In Section 4.2 the model for generating $\boldsymbol{X}$ and $\boldsymbol{y}$ as well as the network structure is presented. Simulation results are shown in Section 4.3 and conclusions and directions for future research are given in Section 4.4.

4.2 System Model

4.2.1 Signal model

We model all targets as a collection of point scatterers. Furthermore we will assume a two dimensional stationary scenario with the radar as the reference point. The received signal from all K scatterers at range r_i and at frequency $f_c + f_m$ can then, after complex down conversion and lowpass filtering, be modeled as [6, p. 529]

$$v(f_m, r_i) \propto \lambda_m e^{j\phi_m} \sum_{k=0}^{K-1} s_k g_{k,i} e^{-j2\pi(f_c+f_m)\frac{2\delta_k}{c}} + w_m, \tag{4.2}$$

where

- $v(f_m, r_i) = v_{m,i}$ has unit Volt

- $\lambda_m = c/(f_c + f_m)$, and accounts for the antenna gain dependence on frequency.
- ϕ_m is a random phase for pulse m.
- f_c is the carrier frequency
- f_m is a deviation from the carrier frecuency in pulse m and is confined to a bandwidth B, i.e. $f_m \in [-B/2, B/2]$
- δ_k is the radial range to scatterer k
- s_k is the amplitude of the k'th scatterer and is proportional to the square root of its RCS and $1/\delta_k^2$
- $g_{k,i}$ is the combined pulse and beamwidth gain at scatterer k and is a function of the difference between the antenna reference angle φ, and the angle to the scatterer, α_k, as well as the difference between δ_k and the sampled range r_i (or in time t_k and t_i). The range r_i is defined as the range to the beginning of range cell i.
- w_m is complex zero mean AWGN with variance σ_w^2.

Let our stepped frequency waveform be such that

$$f_m = \frac{N-1}{\tau'}\left(\frac{m}{M-1} - \frac{1}{2}\right) \tag{4.3}$$

for pulses $m = 0, ..., M-1$, where N determines the total bandwidth $B \approx N/\tau'$, $M \geq 2$, $M \geq N$, and τ' is the full pulse duration of the pulse, not the 3 dB pulse width. Inserting (4.3) in (4.2) and multiplying by $1/\lambda_c$ gives

$$v_{m,i} \propto \beta_m e^{j\phi_m} \sum_{k=0}^{K-1} s_k g_{k,i} e^{-j2\pi\alpha_m \frac{\delta_k}{\Delta r}} + w_m \tag{4.4}$$

where

$$\beta_m = \frac{1}{1 + f_m/f_c} = \frac{1}{1 + \frac{\lambda_c}{2\Delta r}\left(m - \frac{M-1}{2}\right)\frac{N-1}{M-1}} \tag{4.5}$$

$$\alpha_m = \frac{2\Delta r}{\lambda_c} + (N-1)\left(\frac{m}{M-1} - \frac{1}{2}\right) \tag{4.6}$$

$$\Delta r = \frac{c\tau'}{2}. \tag{4.7}$$

This formulation allows us to generate $v_{m,i}$ with all distances normalized to the carrier wavelength, i.e. with $\bar{\delta}_k = \delta_k/\lambda_c$ and $\Delta\bar{r} = \Delta r/\lambda_c$. As for the gain function $g_{k,i}$ we will use a raised cosine in range, representing the received baseband pulse shape; and we model the one way amplitude gain in azimuth with the normalized sinc function such that

$$g_{k,i} = g_\varphi^2(\varphi_k) g_\nu\left(\nu_{k,i}\right), \tag{4.8}$$

where

$$\nu_{k,i} = \frac{\delta_k - r_i}{\Delta r} = \frac{t_k - t_i}{\tau'} \tag{4.9}$$

$$g_\varphi(\varphi) = \begin{cases} \text{sinc}\left(\hat{\varphi}\frac{0.8858}{B_a}\right) \\ \hat{\varphi} = (\varphi + 180^\circ \mod 360^\circ) - 180^\circ \end{cases} \tag{4.10}$$

$$g_\nu(\nu) = \begin{cases} \frac{1}{2}\left[\cos\left(2\pi\nu - \pi\right) + 1\right], & \text{if } 0 \le \nu \le 1. \\ 0, & \text{otherwise,} \end{cases} \tag{4.11}$$

where the angle φ has unit degrees and B_a is the one way antenna half power beamwidth. Letting $\nu = t/\tau'$ for $0 < t < \tau'$ we obtain the frequency response of $g_\nu(\nu)$ as

$$|G_\nu(f)|^2 = \frac{\tau'^2}{4}\text{sinc}^2\left(\frac{\tau' f}{2}\right)\left(\frac{\cos\left(\frac{\pi\tau' f}{2}\right)}{1 - f^2\tau'^2}\right)^2 \tag{4.12}$$

with a 3 dB bandwidth of $1.44/\tau'$. Provided that $M \ge N$, the total bandwidth over all M pulses can be approximated as

$$B \approx \frac{N - 1 + 1.44}{\tau'}. \tag{4.13}$$

Finally, we need a method of generating the point scatterers strength and position, s_k, δ_k, and α_k. Accurately modeling specific targets is by no means a trivial task; as an example, see [95]. Here we will instead model targets as arbitrary two dimensional clouds of point scatterers, of which a standard Rayleigh target can be considered a special case, [86, p. 75-78]. We want to train the neural network to increase the range resolution. Since we cannot expect the resolution to be greater than (4.1), we should generate samples with at least this amount of spacing between targets. For B in (4.13), this yields a spacing between targets of at least $d_{min} \approx \Delta\bar{r}/N$.

Let the total range window of interest be $[\bar{r}_0, \bar{r}_0 + \Delta\bar{R}]$, where $\Delta\bar{R} = \Delta R/\lambda_c \ge \Delta\bar{r}$. The chosen procedure for generating s_k, δ_k, and α_k is then as follows:

1. Draw an integer number of targets $N_T \in [1, N_t^{max}]$ with uniform probability.
2. Draw N_T slots, d_j, $j = 1, ..., N_T$, from the uniform distribution $\mathcal{U}(0, 1)$ and normalize and scale such that $\sum_{j=1}^{N_T} d_j = \Delta\bar{R}$, and ensure that the smallest d_j is at least two d_{min} wide.
3. For each target, draw a target width $q_j \sim \mathcal{U}(0, d_j - d_{min})$ and then draw a target position within slot $p_j \sim \mathcal{U}(d_{min}/2 + w_j/2, d_j - w_j/2 - d_{min}/2)$.
4. For each target, draw an integer number of scatterers $\kappa_j \in [1, \kappa^{max}]$ with uniform probability. The total number of scatterers is then $K = \sum_{j=1}^{N_T} \kappa_j$.
5. For each target, draw scatterer positions within width q_j, $\delta''_{j,i} \sim \mathcal{U}(-w_j/2, w_j/2)$ for $i = 1, ..., \kappa_j$. The total distance to the i'th scatterer in the j'th target is then $\delta'_{j,i} = p_j + \delta''_{j,i} + \sum_{t=1}^{j-1} d_t$. We then relabel all $\delta'_{j,i}$ with indices $k = 1, ..., K$ and let $\delta_k = \bar{r}_0 + \delta'_k$.
6. For each scatterer, draw an angle $\alpha_k \sim \mathcal{U}(-B_a/2, B_a/2)$.
7. For each target, draw an average strength $\bar{s}_j \sim \mathcal{U}(\bar{s}_{min}, \bar{s}_{max})$, and for each scatterer draw a strength $s'_k \sim \mathcal{U}(s'_{min}, s'_{max})$ and let $s_k = \bar{s}_j s'_k \eta_k$, where $\eta_k = \frac{1}{(1+\delta'_k/\bar{r}_0)^2}$ is a normalized range dependent amplitude gain.

The chosen values of all parameters presented in steps 1) - 7) are shown in Table 4.1. This completes our model for generating a signal

$$\begin{aligned} \boldsymbol{x}_i &= [|v_{0,i}|^2, ..., |v_{M-1,i}|^2]^T \\ &= [x_{0,i}, ..., x_{M-1,i}]^T. \end{aligned} \tag{4.14}$$

4.2.2 The Label

We now consider the required label vector $\boldsymbol{y}_i$. We formulate a representation of all scatterers as an N-point vector $\boldsymbol{y}_i$ with n'th element given by

$$y_{n,i} = \sum_{k=0}^{K-1} \frac{(s_k g_{k,i})^2}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(n + \frac{1}{2} - \nu_{k,i} N\right)^2}{2\sigma^2} \right) \tag{4.15}$$

for $n = 0, ..., N-1$. This Gaussian kernel filtered version of the scattering points will have a resolution that depends on σ^2. With $\nu = t/\tau'$, the 3 dB bandwidth of the Gaussian kernel is

$$B_G = \frac{\sqrt{\ln(2)}}{\pi\sigma} \frac{N}{\tau'}. \tag{4.16}$$

Table 4.1: Parameter Values

Parameter	**Description**	**Value**
$\Delta\bar{r}$	Pulse length in meters normalized by carrier wavelength	6700
M	Number of pulses	64
N	Samples within pulse in neural network output	32
B	Total bandwidth, determined by N and $\Delta\bar{r}$	$0.0024 f_c$
SNR	Signal to noise ratio	-3 to 30 dB
ε	Range sampling interval in pulse lengths	1/8
N_r	Range samples in neural network input	5
N'	Number of synthetic range samples in neural network output, $\mathbf{y}$, determined by N, N_r and ε	48
N_t^{max}	Maximum number targets	10
κ^{max}	Maximum number of scatterers per target	100
$[\bar{s}_{min}, \bar{s}_{max}]$	Interval of "average" strength per target	[0.1, 1]
$[\bar{s'}_{min}, \bar{s'}_{max}]$	Interval of initial strength per scatterer	[0.1, 10]
B_a	Antenna 3 dB beamwidth, one way/two way	3° / 2.2°

By setting $\sigma^2 = \ln(2)/\pi^2$ we set the kernel bandwidth approximately equal to the full bandwidth in (4.13), which gives the theoretical range resolution in (4.1).

Another choice for $\boldsymbol{y}_i$ could be the result obtained from a coherent system, i.e. when the phase ϕ_m in (4.2) is known for each pulse. This has a trivial solution to the proposed problem. Assume all scatterers are confined to an interval of one pulse length such that

$$\delta_k = r_i + \frac{n + \Delta n_p}{N} \Delta r \tag{4.17}$$

for some integer $n \in [0, N-1]$ and Δn_p is uniformly distributed in the interval $(0, 1]$. We then re-label all s_k with index n, p, i.e. $s_k \to s_{n,p}$ for $p = 0, ..., P_n - 1$, where P_n is the number of scatterers in the interval $[n, n+1]$. Ignoring noise, we can then express (4.4) as

$$v_{m,i} \propto \beta_m e^{-j\theta_{m,i}} \sum_{n=0}^{N-1} \tilde{y}_{n,m,i} e^{-j2\pi(m-m_0)\frac{N-1}{M-1}\frac{n}{N}} \tag{4.18}$$

where

$$\tilde{y}_{n,m,i} = \sum_{p=0}^{P_n-1} s_{n,p} g_{n,p,i} e^{-j2\pi(m-m_0)\frac{N-1}{M-1}\frac{\Delta n_p}{N}} \tag{4.19}$$

$$\theta_{m,i} = 2\pi \left(\frac{m(N-1)}{M-1} - m_0 \right) \frac{N-1}{M-1} \frac{r_i}{\Delta r} - \phi_m \tag{4.20}$$

$$m_0 = \frac{M-1}{2} - \frac{2\Delta r(M-1)}{\lambda_c(N-1)}. \tag{4.21}$$

For $M = N$ and using (4.14) we can write (4.18) as

$$\boldsymbol{x}_i \propto \left| \text{diag} \left(\boldsymbol{F}_{m_0} \tilde{\boldsymbol{Y}}_i \right) \right|^2 \tag{4.22}$$

where $\boldsymbol{F}_{m_0} \in \mathbb{C}^{N\times N}$ is the N-point DFT matrix shifted in frequency by m_0 and $\tilde{\boldsymbol{Y}}_i \in \mathbb{C}^{N\times N}$, with elements $\tilde{\boldsymbol{Y}}_n[m,i] = \beta_m e^{-j\theta_{m,i}} \tilde{y}_{n,m,i}$. We can thus approximate $\boldsymbol{x}_i$ as the squared magnitude of the N-point DFT of $\tilde{\boldsymbol{y}}_i = [\tilde{y}_{0,m,i}, ..., \tilde{y}_{N-1,m,i}]$ rotated by $\theta_{m,i}$ and modulated by m_0.

Regarding the elements of $\tilde{\boldsymbol{y}}_i$, we note that these depend on the frequency index m, and by taking the IDFT of (4.18) we obtain an approximation of $\tilde{\boldsymbol{y}}_i$ with undesired artifacts. This makes it less interesting as a training label. We will however use $|\tilde{\boldsymbol{y}}_i|^2$ as a reference when evaluating our results. Since $|\tilde{\boldsymbol{y}}_i|^2$ spans a range $\Delta\bar{r}$, the resolution of $|\tilde{\boldsymbol{y}}_i|^2$ is $\Delta\bar{r}/N$, an increase in resolution of a factor $N/1.44$ due to the cosine shape of the pulse. To distinguish between range resolution before and after pulse compression we will use the term "synthetic range cells" after pulse compression and "range cells" before pulse compression. Note that over several range cells $i = 1, ..., N_r$ the expression in (4.18) can be approximated as a short time Fourier transform (STFT) where the pulse shape acts as the window function.

As the aspect angle changes, the radial positions of all scatterers will change slightly giving rise to the well known glittering effect, or target fluctuations, often modeled with one of four Swirling models (Rayleigh Targets) [30].

4.2.3 Ambiguities and the SNR

The uniqueness of an inverse phase problem applies up to so called $trivial\ ambiguities$ [88], since we can always write

$$|\boldsymbol{G}(f)| = |\mathcal{F}\{g(t)\}| = |\mathcal{F}\{g(t-\tau_0)\}| = |\mathcal{F}\{g(-t)\}|.$$

We can therefore not expect the network to learn the correct shift or reflection of $\boldsymbol{y}_i$. This issue is encountered in any phase retrieval problem and the only way to solve it without prior knowledge of $\boldsymbol{y}_i$, is to correlate or match overlapping estimates $\hat{\boldsymbol{y}}_i$ and $\hat{\boldsymbol{y}}_{i+1}$ where we have sampled the range with period $0 < T_s < \tau'$.

This is exploited in the iterative Griffin-Lim algorithm for phase retrieval [96], which is based on the STFT. The known overlap of samples and window shape acts as the only prior knowledge. If additional prior knowledge is known a wide range of phase retrieval algorithms exist for the STFT [97]. We look closer at the Griffin-Lim algorithm in Section 4.3.4. As with the Griffin-Lim algorithm, we will feed the network with N_r consecutive range samples of $x_{m,i}$, for $i = 0, ..., N_r - 1$, at each pulse m, yielding a STFT "block", and leave the process of exploiting overlapping information to the network. The range sampling is such that

$$r_i = r_0 + i\varepsilon\Delta\bar{r} \tag{4.23}$$

where $\varepsilon = T_s/\tau' \in [0, 1]$. From (4.14), we form $\boldsymbol{X} \in \mathbb{R}^{M\times N_r}$

$$\boldsymbol{X} = [\boldsymbol{x}_0, ..., \boldsymbol{x}_{N_r-1}] \tag{4.24}$$

covering a range

$$r_X = \Delta\bar{r}(1 + (N_r - 1)\varepsilon). \tag{4.25}$$

Each element of our label $\boldsymbol{y}$ is

$$y_n = \frac{1}{N_r}\sum_{i=0}^{N_r-1} y_{n,i} \tag{4.26}$$

for $n = 0, ..., N'$ where $N' = N(1 + (N_r - 1)\varepsilon)$. For practical reasons we will choose N and ε such that $N\varepsilon$ is an integer. As explained in Section 4.2.4 we have set $M = 2N$, and as stated in [98], the inverse problem is then equivalent to recovering $\boldsymbol{y}$ from its autocorrelation function,

Regarding the SNR, we only consider the range cell with the highest power when setting the SNR of a target. A target may or may not span several range cells. The SNR is defined as

$$\text{SNR} = \max_i \frac{1}{M}\frac{\sum_{m=0}^{M-1}\tilde{x}_{m,i}}{\sigma_w^2} \tag{4.27}$$

where $\tilde{x}_{m,i}$ is the noise free version of $x_{m,i}$.

4.2.4 Network model

The input data $\boldsymbol{X}$ and label data $\boldsymbol{y}$ are first normalized in order to reduce the required dynamic range of the network and ease the training process. This normalization also includes logarithmic scaling and is done by setting

$$\bar{p} = \frac{1}{N_r} \sum_{i=0}^{N_r-1} \sum_{m=0}^{M-1} x_{m,i} \tag{4.28}$$

$$x'_{m,i} = \log\left(\frac{M x_{m,i}}{\bar{p}}\right) \tag{4.29}$$

for $m = 0, ..., M-1$ giving $\boldsymbol{X}' = [\boldsymbol{x}'_0, ..., \boldsymbol{x}'_{N_r-1}]$ a mean value of one in linear scale. We also normalize the label in the same manner. Since many elements of $\boldsymbol{y}$ may have value zero, we need to add a small bias and therefore set

$$y'_n = \log\left(\left(\frac{N' y_n}{\sum_{n=0}^{N'-1} y_n} + 0.001\right)\frac{1}{1.001}\right) \tag{4.30}$$

giving $\boldsymbol{y}'$ a mean value of one in linear scale. The choice of this bias affects the network performance, and a different value may be more suitable for a different signal model. Our final estimate, $\hat{\boldsymbol{y}}$, is then obtained as

$$\hat{\boldsymbol{y}} = \frac{\bar{p}}{N'}\left(1.001\exp(\hat{\boldsymbol{y}}') - 0.001\right) \tag{4.31}$$

where $\hat{\boldsymbol{y}}' \in \mathbb{R}^{N'\times 1}$ is the CNN output.

A sample $\boldsymbol{X}$ containing only noise will have a randomness consistent with a spatially large target with many point scatterers. After normalization these two will have similar appearance. We can thus approximate the mean output of the CNN for a noise only sample by making (4.15) continuous in ν_k and setting s_k to some arbitrary constant, e.g.

$$y_{n,i}^{\text{noise}} = \int_{-\infty}^{\infty} \frac{g_\nu^2(\nu - \tilde{r}_i)}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(n + \frac{1}{2} - (\nu - \tilde{r}_i)N\right)^2}{2\sigma^2}\right) d\nu \tag{4.32}$$

where we have ignored the azimuth gain and $\tilde{r}_i = r_i/\Delta r$. We then use (4.26) to obtain y_n^{noise} from (4.32) and the noise power as

$$\bar{P}_{\text{noise}} = \frac{y_{N'/2}^{\text{noise}}}{\sum_{n=0}^{N'-1} y_n^{\text{noise}}} \tag{4.33}$$

where we estimate the noise power at the midpoint $n = N'/2$, or peak of this noise "label." We can do the same calculation for the label of a single point scatterer and take the ratio of the peak of the midpoint and $\bar{P}_{\text{noise}}$ to obtain the gain in SNR. Note that this assumes that the label/CNN output is converted to linear scale. With the chosen parameter values this results in a gain of 13 dB. For low SNRs this approximation will not hold as the noise will result in a widening of the synthetic pulse shape, causing decreased gain in SNR due to the normalization in (4.28) and (4.30). This also assumes a good fit to the label after training, and therefore this gain in SNR is only approximate.

The CNN consists of two parts. The first part has four convolutional layers where each of the N_r columns in the input are transformed to a 16 element vector by the same set of weights. The second part has four fully connected layers transforming the $N_r \times 16$ data point to the $N' \times 1$ estimate $\hat{\boldsymbol{y}}'$. The motivation for this setup follows from the reasoning in Section 4.2.3, i.e. in the first part we transform the data to time domain and in the second we correlate and align the N_r samples. This is at least the motivation. Whether this is what is actually happening or not is a different question.

According to [99] the number of frequency samples required for phase retrieval is $M \geq 2N - 1$ and we therefore set $M = 2N$. This applies to real valued $\boldsymbol{y}$, and as mentioned $\boldsymbol{y}$ is complex in this case, for which the required M is actually $M \geq 4N - 2$. Since we are only interested in the magnitude of $\boldsymbol{y}$ we still choose $M = 2N$. Experiments were performed with $M = 4N$ and gave comparable results as $M = 2N$. Experiments were also performed with $M = N$ and gave worse performance than $M = 2N$, see Section 4.3.1.

As for ε in (4.23), and the number of range samples N_r in $\boldsymbol{X}$, a grid search was performed over a range of values. As could be expected the performance improved with larger N_r and a value of $N_r = 5$ was finally chosen. With sampling interval $\varepsilon = 0$ we obtain a repetition of the same samples and gain no information. With $\varepsilon = \tau'$ we have no overlap of information in two consecutive range cells. It is therefore not surprising to find an optimal value within this range. With the chosen setup, this was $\varepsilon = 4/32$.

From our spacing between targets constraint, d_{min}, we are restricted to set $N_t^{max} \leq N/2$. However for $N_t^{max} = N/2 = 16$ we have deterministic positioning of targets which is in this case undesired as we do not want the network to be biased to any particular target configuration. For this reason the maximum number of targets was limited to 10. All other parameter values were arbitrarily chosen. All distances are normalized to wavelengths. As an example, for an X-band radar at $f_c = 10$ GHz these numbers would correspond to a system with full pulse length Δr = 200 m, 24 MHz bandwidth and 25 m interval in range sampling.

The final network architecture is presented in Table 4.2. The values of the

Table 4.2: Neural Network Architecture

Layer Type	Weight Shape	Output Shape	Activation
Convolutional	F: 64, K: 16×1	$64 \times 49 \times 5$	ReLu
Convolutional	F: 32, K: 1×1	$32 \times 49 \times 5$	ReLu
Convolutional	F: 16, K: 1×1	$16 \times 49 \times 5$	ReLu
Convolutional	F: 16, K: 49×1	$16 \times 1 \times 5$	ReLu
Fully Connected	48×80	48×1	ReLu
Fully Connected	48×48	48×1	ReLu
Fully Connected	48×48	48×1	ReLu
Fully Connected	48×48	48×1	Linear

F: Number of filters/kernels. K: Kernel size. All Convolutional layers had stride [1,1] and no zero padding. Total number of parameters: 27200.

kernel sizes and number of layers are a result of empirical testing, as is often the case with neural networks. A deeper network did yield slightly lower validation loss at the expense of longer execution time. Since a motivating factor to choose an ANN based solution is the relatively fast and deterministic computation time, we aimed for a small network that still gave satisfying results.

The gain beyond the eight layers in Table 4.2 was subtle with a training/validation loss of 6.36/6.36 for 8 layers and 6.33/6.33 when adding one additional fully connected layer of the same size. Subtracting one of the four fully connected layers gave a training/validation loss of 6.43/6.43, trained on 10^6 samples for 10 epochs. Network architecture optimization is of course more complicated than simply adding layers. As with depth, a wider network will in general yield lower loss at the expense of computation time. As an example, increasing the output dimension of the first three fully connected layers to 100 instead of 48 gave a validation loss of 6.06 instead of 6.36, however this also entails an increase in parameters by over 80%. The chosen filter length of 16 in the first convolutional layer has been optimized with regard to the chosen distribution of SNRs. If only high SNRs were considered a longer filter tended to yield lower loss. As described in Section 3.3, the network was trained using the empirical mean squared error loss with stochastic gradient descent and Adam optimisation.

4.3 Results

Results on training the CNN are presented in Section 4.3.1. As mentioned in the introduction, pulse compression is used to increase range resolution and the SNR. Results on range resolution are presented in Section 4.3.2 and results on

Table 4.3: Batch Size and Loss

Batch Size	Training Loss*	Validation Loss*	Execution Time†
32	6.20	6.35	14 s
64	6.23	6.38	8 s
128	6.27	6.38	4 s
256	6.45	6.52	3 s
512	6.46	6.54	2 s

*After 100 epochs with 10^5 training samples and 10^4 validation samples.
†Time per epoch on an Nvidia Geforce RTX 2080 GPU.

SNR gain are presented in Section 4.3.3. Although we have limited this study to stationary scenarios, we also present some results on non-stationary cases as well as a comparison to other methods in Section 4.3.4.

4.3.1 Training the CNN

Five different batch sizes were tested, the results of which are shown in Table 4.3. Based on this, a batch size of 128 was chosen since it gave a good trade-off between execution time, overfitting, and low loss.

Table 4.4: Number of Pulses M and Loss

M	Training Loss*	Validation Loss*	Network Size†
32	6.54	6.64	F:[64,32,33,20] K:[16×1, 1×1, 1×1, 17×1] #P: 27401
64	6.27	6.38	F:[64,32,16,16] K:[16×1, 1×1, 1×1, 49×1] #P: 27200
128	6.22	6.34	F:[16,10,10,16] K:[32×1, 1×1, 1×1, 97×1] #P: 27288

*After 100 epochs with 10^5 training samples and 10^4 validation samples. †F=Number of filters/kernels, K=Kernel sizes in the first 4 convolutional layers. #P=number of parameters.

Fig. 4.1 shows the loss as a function of batch iterations for four different sizes of training data. The same validation set with 10^4 samples was used in all four cases. All datasets are subsets of any larger dataset, where the full dataset has 10^6 samples. Significant overfitting can be seen when the number of training samples

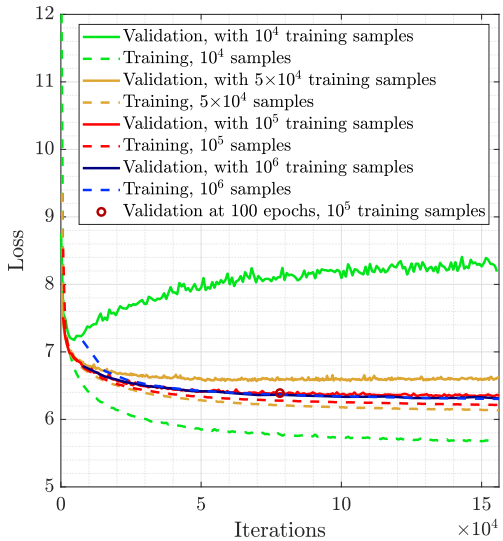


Figure 4.1: Training and validation loss for different sizes of training data. The network was trained on each dataset an equal amount of iterations with a batch size of 128. The red circle marks the validation loss at 6.4 for the selected training process with 10^5 training samples and 100 epochs of training. The same validation dataset with 10^4 samples was used in all four cases.

is 10^4 and virtually no overfitting occurs with 10^6 samples. A sample size of 10^5 and 100 epochs of training was chosen since the validation loss reaches the same level when using 10^6 samples and decreases negligibly with further training.

In order to make the network more robust to noise, the SNR of the training and validation data was uniformly distributed between -3 and 30 dB in logarithmic scale. The mean loss depends on this distribution, and there is a slight trade-off between minimizing loss and SNR robustness. A network trained only on data with 30 dB SNR achieved a validation loss of 5.4, and 10.5 when evaluated on data with 3 dB SNR. Conversely, a network trained only on data with 3 dB SNR converged to a validation loss of 7.4, and 16.5 when evaluated on data with 30 dB SNR. A network trained on the proposed range of SNRs gave a loss of 7.5 on data with 3 dB SNR and 5.6 on data with 30 dB SNR. The networks performance at a specific SNR does therefore not appear to suffer too much in terms of loss despite having been trained on a broader distribution of SNRs.

Tests were performed with $M = N$, $M = 2N$ and $M = 4N$, where $N = 32$ is the number of synthetic resolution cells within one pulse after pulse compression.

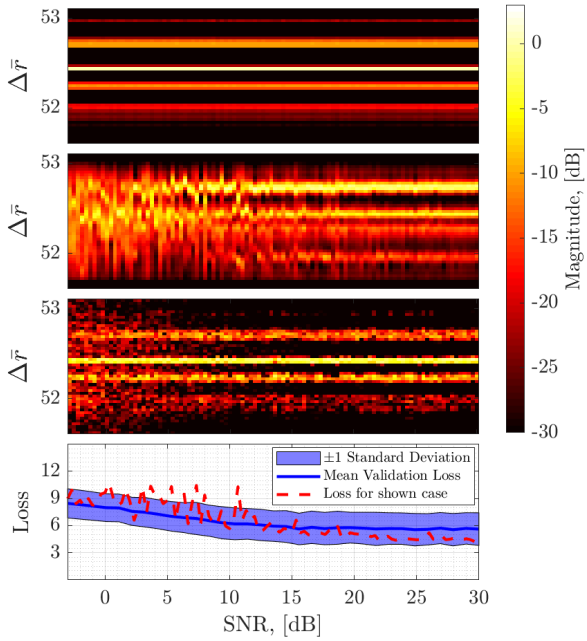


Figure 4.2: Example of improvement with SNR for a five target scenario. The range to the first target is 52 $\Delta\bar{r}$ (10.4 km with f_c =10 GHz). The top plot shows the ground truth. The second is the CNN estimate as a function of SNR. The third plot is the coherent IDFT estimate, shown for reference. The bottom plot shows the loss at each SNR for both this specific case and the mean validation loss ± 1 standard deviation. Below 10 dB SNR the resolution of the CNN drops significantly in the second plot, which correlates well with the loss in the bottom plot.

The number of filters and kernel sizes in the first four convolutional layers were modified such that the total numbers of parameters were comparable and the output shape from the last convolutional layer was always $16\times1\times5$. The loss for $M = 128$ was comparable to the loss when $M = 64$ while the loss for $M = 32$ was slightly higher, see Table 4.4. For this reason we chose $M = 64$.

The mean training loss for the chosen solution was 6.3 and the mean validation loss was 6.4. The validation loss as a function of SNR is shown in the bottom plot in Fig. 4.2. Since the loss is taken on $\boldsymbol{y}'$ in logarithmic scale, it has little physical meaning. For reference, the linear minimum MSE solution of $\boldsymbol{y}' = \boldsymbol{W}\boldsymbol{z}$ with vectorized $\text{vec}(\boldsymbol{X}') = \boldsymbol{z}$, had average loss $L \approx 10$ for both training and validation

data. We note in Fig. 4.2 that the loss increases for SNR$<$10 dB. To further visualize this, Fig. 4.2 shows a case with five targets within one $\Delta\bar{r}$. For higher SNR values, the four closest targets are clearly resolved with the CNN solution (second plot from top).

4.3.2 Resolution Gain

It is common to specify the range resolution of a pulsed compressed radar in terms of the peak-to-sidelobe ratio of the synthetic pulse, i.e. the pulse shape after pulse compression. In this case however we have a highly non-linear neural network functioning as the pulse compression "filter", and conclusions drawn from such metrics may not apply in the same way as in linear systems. Instead we are left to numerical evaluations where we evaluate the CNN over a broad range of simulated scenarios.

We begin with what can be regarded as a synthetic pulse shape by looking at the return of a single point scatterer with no noise. This is shown in Fig. 4.3. As in [100] we will benchmark the performance against an equivalent coherent system. The coherent synthetic pulse shape is also shown in Fig. 4.3 with and without windowing before the IDFT. Without windowing we obtain the sinc-pattern with 13.3 dB sidelobes. For our comparisons we use the windowed version that has a synthetic pulse shape similar to the one obtained with the CNN.

Although the CNN pulse shape does not have sidelobes in the same way, it is interesting to note that there are similar patterns of "peaks" at approximately the same locations as the coherent IDFT based estimate, at positions 0.5,1.5,...,47.5. These are the result of the network optimization and were not present in the label $\boldsymbol{y}$. The "sidelobe" levels of the CNN are at a rather constant level and below -35 dB. For this reason, a Taylor window was used with -35 dB sidelobe suppression over $\bar{n} = 50$ lobes, see Section 2.18. This results in a widening of the 3 dB synthetic pulse width by a factor 1.2, making it closer to the apparent shape of the CNN based synthetic pulse shape.

The 3 dB synthetic pulse width from the CNN was measured to about one synthetic range cell. Note that for the coherent case we only need samples from one range cell to obtain the N-point synthetic range, i.e. one $\boldsymbol{x}_i$ for some i while the DNN requires $N_r = 5$ consecutive range samples. For the sake of comparison we let the coherent IDFT estimate be the average of $N_r = 5$ overlapping samples of $\boldsymbol{x}_i$. It thereby uses the same input data, $\boldsymbol{X}$, as the non-coherent CNN solution and produces the same output size N'.

To quantify the resolution, we use the peak-to-valley ratio which we define as

$$\frac{\min(P_1, P_2)}{V} \tag{4.34}$$

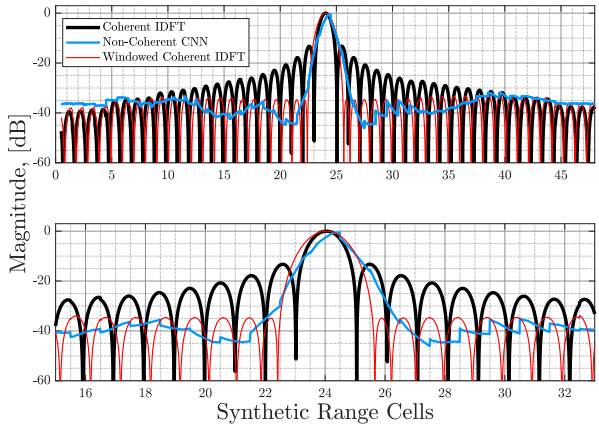


Figure 4.3: Numerically evaluated synthetic pulse shape over all $N' = 48$ synthetic range cells. The coherent solution using the IDFT is also shown for reference without windowing, yielding the characteristic sinc pattern, and with a Taylor window with S=-35 and $\bar{n} = 50$.

where P_1 and P_2 are the power levels of two consecutive peaks/targets and V is the "valley" power estimated from the signal power between the most adjacent scatterers of the two targets. This was estimated as the mean of the n smallest synthetic range cells out of a maximum of 5, where $1 \leq n \leq 3$ depending on the distance between the targets. This is illustrated in Fig. 4.4. Note that the range covered in Fig. 4.4 is two full pulse lengths, equivalent to 64 synthetic range cells. This was obtained by averaging several estimates with a sliding window approach over range cells with stride equal to one.

In plot (a) in Fig. 4.5 we estimate the peak-to-valley ratio for two single point scatterers of equal size. The average peak-to-valley ratio over 1000 Monte Carlo runs is shown as a function of distance between targets in synthetic range cells. Table 4.5 shows the minimum distance in synthetic range cells required for the peak-to-valley ratio to be at least 3 dB with 90% and 95% empirical probability. For SNRs 3 dB and 7 dB this probability was never reached with the CNN. Note that these incorporate losses due to interference between targets as well as noise. Had it not been for these factors, two point scatterers should be separable with 3 dB margin after a separation of about one synthetic range cell with coherent pulse compression.

The results in plot (a) in Fig. 4.5 and Table 4.5 are only for two single point scatterers. Similar simulations were performed using samples drawn from the

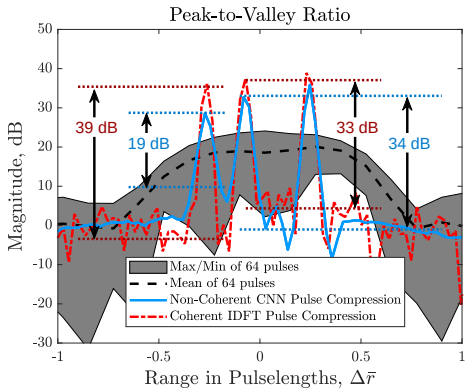


Figure 4.4: Example of peak-to-valley ratios for a scenario with three targets. For reference, the mean pulse shape before pulse compression is shown as the black dashed line, as well as the maximum and minimum values of the 64 pulses. The SNR was 20 dB.

model presented in Section 4.2.1, from which the training and validation data was generated, but with slight modifications. The number of targets generated were between two and ten (not one and ten) and the peak-to-peak ratio of two consecutive targets defined as

$$|10\log_{10}(P_1/P_2)| \tag{4.35}$$

using the power levels of the coherent IDFT estimate, was at most 10 dB. Note that the specified SNR is before pulse compression and is thus an average over all targets within one pulse. The corresponding results for these tests are shown in plot (b) in Fig. 4.5 and Table 4.6. Looking at Table 4.5 and 4.6 we see that for SNR 10 dB and above, the resolution is approximately a factor 2 worse in synthetic range than the coherent solution using this metric. We also note in plot (b) in Fig. 4.5 that the median peak-to-valley ratio of the non-coherent CNN solution is approximately at the same level as the coherent peak to valley ratio when SNR$\geq$10 dB and the target separation exceeds 10 synthetic range cells.

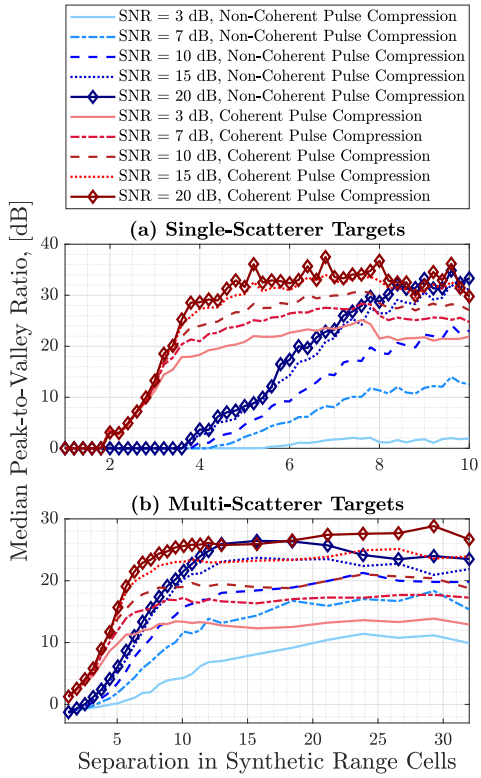


Figure 4.5: Empirical median peak-to-valley ratio as a function of the separation between targets in synthetic range for different SNRs. The median was taken from 1000 runs. The distribution of peak-to-valley ratios was quite skewed. For this reason the median was used as the average rather than the mean. (a) Two point scatterers of equal size. The peak-to-valley ratio begins increasing after a separation of about two synthetic resolution cells for coherent pulse compression and about four synthetic cells for the non-coherent pulse compression, depending on SNR. (b) Multiple targets with multiple scatterers, drawn from a modified version of the model in Section 4.2.1. The modifications included the number of targets to be at least two and the relative peak-to-peak ratio was no more than 10 dB after coherent pulse compression. Although the non-coherent pulse compression requires a larger separation between targets than coherent pulse compression, it can still reach the same levels if the SNR is sufficiently high. Note that the SNR is before pulse compression and includes the power of all targets. The separation in synthetic range refers to the distance between the end and beginning of two consecutive targets.

Table 4.5: Two Target Required Range† Separation for ≥3 dB Peak-to-Valley Ratio with **90%** and 95% Empirical Probability

SNR [dB]	Non-Coherent CNN*	Coherent IDFT*
20	**4.8**/5.6	**2.8**/3
15	**5**/5.8	**2.8**/3
10	**6.6**/7.2	**2.8**/3
7	-/-	**2.8**/3
3	-/-	**2.8**/3.4

†Separation in synthetic range cells for two point scatterers. *Bold number corresponds to **90**% and plain number corresponds to 95%.

Table 4.6: Multiple Target Required Range† Separation for ≥3 dB Peak-to-Valley Ratio with **90%** and 95% Empirical Probability

SNR [dB]	Non-Coherent DNN*	Coherent IDFT*
20	**7.4**/9.4	**6.2**/7.8
15	**7.8**/9.8	**6.2**/8.2
10	**9.8**/11.8	**6.6**/9.4
7	**11.4**/17.4	**6.6**/9.4
3	**-**/-	**7**/9.4

†Separation in synthetic range cells for multiple multi-scatter target scenarios. *Bold number corresponds to 90% and plain number corresponds to 95%.

In Fig. 4.6 we show an example with several combined estimates of $\boldsymbol{y}$ that form a B-scope view. The scan rate and PRF were chosen such 64 pulses corresponds to 2°. We then sample 25 range cells and scan 20°. The CNN estimate is formed by sliding a window over the sampled data with size 5×64 and stride [1,64], yielding an output estimate $\hat{\boldsymbol{y}}_i \in \mathbb{R}^{48\times 1}$ for $i = 0, ..., 20$. Two estimates $\boldsymbol{y}_i$ and $\boldsymbol{y}_{i+1}$ then overlap by $48 - 4 = 44$ samples and the estimates in Fig. 4.6 show the mean of all overlapping samples. For reference, we also show the same results from the coherent IDFT estimate. We see in Fig. 4.6, that the three targets can be resolved despite the loss of phase information using the CNN.

4.3.3 SNR Gain

In this section we will consider the gain in SNR. Fig. 4.7 shows the mean SNR gain for single point scatterers as a function of SNR as well as an intensity map to illustrate the distribution in gain. The empirical 5th percentile for the coherent gain was 19.5 dB for SNR≥0 dB. For the non-coherent gain and SNR≥ 5 dB, the

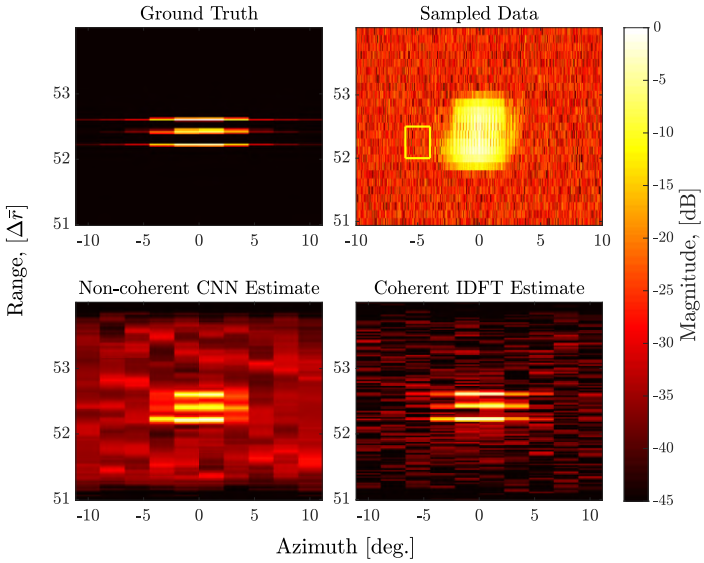


Figure 4.6: Example of a simulated scan with three targets positioned within one pulse length $\Delta\bar{r}$, at 52 pulse lengths range. With $f_c = 10$ GHz this corresponds to 10.4 km ($\Delta r = 200$ m) requiring a pulse repetition frequency $<$ 14kHz. The scan has 64 pulses in 2°, which would then require a scan rate less than $360°/0.83$s for a mechanical antenna system. The SNR in $\boldsymbol{X}$, top right plot, was 20 dB at the targets. The bottom right plot is for reference and shows the coherent result using the IDFT. The yellow box in the top right plot marks the input size to the CNN. As seen, the three targets can be resolved despite the loss of phase information.

5'th percentile gain was greater than 6 dB and the mean gain was greater than 12 dB, consistent with the 13 dB calculated in Section 4.2.4. In terms of the mean, this is a loss of 10 dB from the coherent case. Fig. 4.8 shows the synthetic pulse shape from a single point scatterer, for various SNRs. The gain in SNR becomes less significant for SNRs below 3 dB.

In Fig. 4.6 we see that the background noise characteristics is different in the coherent and non-coherent estimates. This will affect the trade-off between false alarm and detection probability, or the ROC. Fig. 4.9 shows estimates of the ROC for a target consisting of five point scatterers with exponentially distributed RCS, and uniformly distributed in a space confined to one synthetic range cell. The scatterer positions and RCS were constant in each sample, or set of 64 pulses, but varied between samples. This gave a distribution of the signal power, or target RCS, that was close to that of a target with log-normal PDF, a common model for target fluctuations from one scan to another. A perhaps more "standard" model is

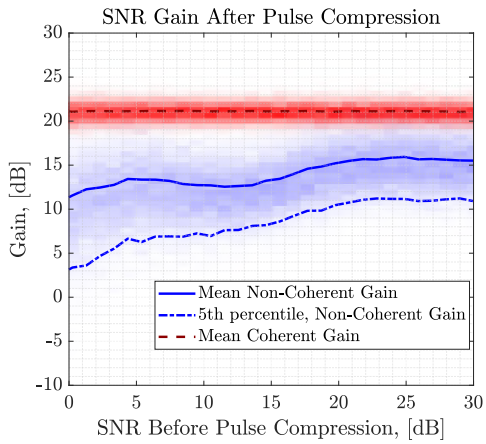


Figure 4.7: Empirical mean gain in SNR as a function of SNR for non-coherent and coherent pulse compression. The coherent pulse gain is a result of both pulse compression as well as the averaging of 5 samples. The SNR after pulse compression was estimated as the power at the target divided by the mean power of 20 noise samples minus one. The sample size was 1000 for each SNR.

the Swerling III model, which has a similar PDF in target power, see bottom plot in Fig. 4.9. Fig. 4.9 shows a comparison of the histogram of 5×10^5 samples of simulated target power and the PDF of a log-normal target scaled to have mass 5×10^5. The ROC estimate is for a constant threshold detector.

In the top plot in Fig. 4.9 we have compared the CNN solution to an alternative form of non-coherent pulse compression using OOK, see Section 2.3. We have compared it to a waveform using m-sequence coding of sub-pulses [44], with the same instantaneous bandwidth as the CNN solution. The length of such an m-sequence is $2^m - 1$ which gives 63 sub-pulses for $m = 6$. After Manchester coding, [44], the total length of the pulse sequence is 126 sub-pulses with 63 "on" pulses, i.e. it uses approximately the same number of pulses as the presented CNN solution but transmitted within one PRI instead of 64 with a single "sub-pulse" in each. We see in Fig. 4.9 that simply integrating 64 pulses yields better results in terms of ROC than the other non-coherent solutions. The CNN solution yields equal or better results than the 6-sequence OOK waveform for SNR greater than 7 dB. A shorter m-sequence will have worse performance, and a longer better in terms of ROC. Note that the 6-sequence OOK solution is obtained from a single pulse and the CNN solution requires 64. On the other hand, the 6-sequence OOK method offers no increase in resolution which is the main objective in this study. Also note that the blind range of such a 6-sequence OOK system is 126 sub-pulses, or ca

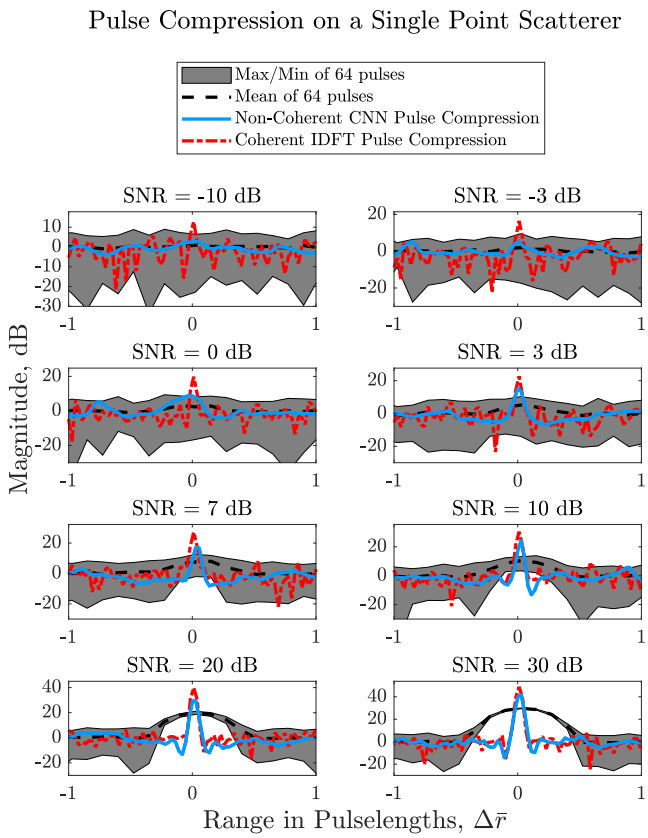


Figure 4.8: Example of compressed pulse for different SNRs of a point scatterer. The mean, as well as maximum and minimum range of the 64 simulated pulses are shown for reference.

25 km if Δr=200m. The coherent estimate had empirical detection probability of approximately one, with false alarm probability of almost zero for the three considered SNRs.

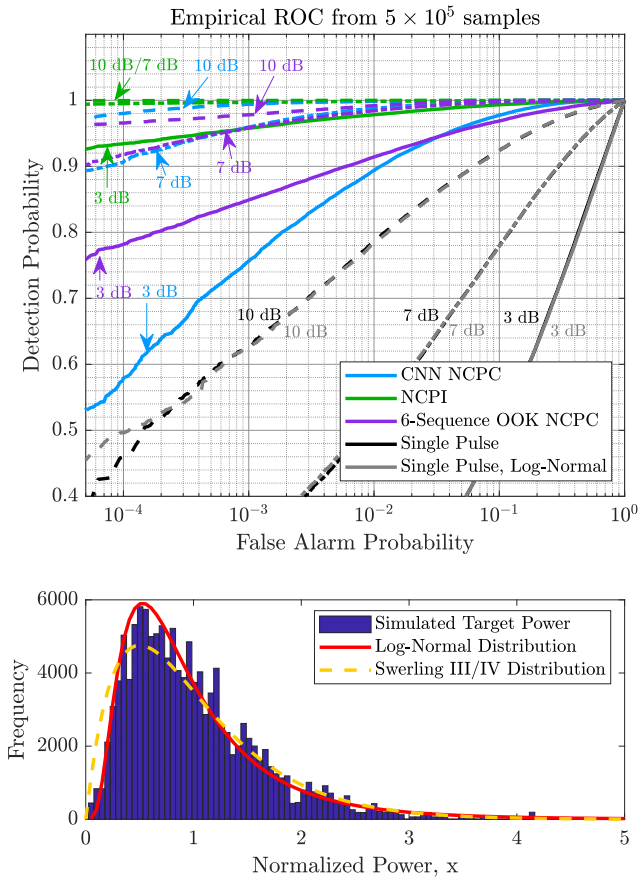


Figure 4.9: ROC curves for different scenarios. The top plot shows the ROC curve estimated from 5×10^5 samples. The curves are for the CNN solution (blue), pulse integration over $M = 64$ pulses (green) single pulse detection (black) and a non-coherent pulse compression method with OOK using an m-sequence with m=6 (purple). The m-sequence was generated from a shift register with initial values [1,1,1,1,1,1]. The bottom plot shows the histogram of the normalized target power levels and the PDF of a log normal distribution and the similar Swerling III/IV PDF (scaled to have the same mass). For reference, ROC curves for single pulse detection on a true Log-Normal fluctuating target in AWGN is also shown in the top plot (grey). These are quite consistent with the corresponding curves of the simulated target. The CNN ROC curves are for single evaluations of the CNN and no averaging obtained from sliding window estimates was performed. The given SNRs are the mean SNRs. NCPC=Non-Coherent Pulse Compression, NCPI=Non-Coherent Pulse Integration.

4.3.4 Other Scenarios and Methods

We saw in Section 4.2.3 that the problem presented in this study could potentially be solved using certain phase retrieval algorithms. Fig. 4.10 shows a comparison between the presented CNN solution and an alternative phase retrieval solution using the Griffin-Lim algorithm (GLA) [96]. This exploits the fact that the N_r columns of $\boldsymbol{X}$ can be regarded as the STFT of the desired signal $\boldsymbol{y}$. The known pulse shape and window interval then acts as prior knowledge necessary for phase retrieval, although this is not always sufficient [97]. By iterating between time and frequency domain and applying what is known in each domain, the error between the estimated $\boldsymbol{y}$ and true $\boldsymbol{y}$ will reduce with each iteration, [96]. In this case we stopped the iteration when this error, measured as the mean square error, changed by less than 10^{-6}. This required 100-600 iterations for the five cases shown in Fig. 4.10, where each case is identical apart from having different pulse shapes.

The pulse shapes are raised cosine filters with different shape parameters $\beta \in [0, 1]$. A value of $\beta = 1$ yields the simple raised cosine in (4.11). Note that the CNN solution performs reasonably well for the whole range of β when N=32, despite having only been trained on $\beta = 1$. Ignoring some false targets, we note that the GLA works reasonably well with rectangular pulses. However, when introducing some bandwidth limitations on the pulse, i.e. increasing β, the GLA fails for $\beta > 0.7$. This type of case specific sensitivity is a well known problem with the GLA. It is discussed in [94] where the authors combine the GLA and a deep neural network (DNN) to solve a phase retrieval problem in order to gain more robust and accurate phase estimates. This is however a semi-iterative solution where each iteration involves an evaluation of a five-layered neural network. The number of iterations used in [94] was 10, equivalent to a network with 50 layers.

In this study we have only considered stationary scenarios up to this point. Although dynamic scenarios are left for future work, we will still evaluate the CNN on a couple examples. Relative motion between target and radar is always a problem when using the stepped frequency waveform due to the fact that M pulses are required, [101]. Assume that the target has moved a distance $d = \Delta r L/(N-1)$ pulse lengths along the radar line of sight during the M pulses, typically $L \ll N$. Using $v_{m,i}$ in (4.4) we can then write the received signal $\bar{v}_{m,i}$ as

$$\bar{v}_{m,i} = \left(\frac{R_0}{R_m}\right)^4 v_{m,i} e^{-j2\pi\alpha_m \frac{mL}{(M-1)(N-1)}} \tag{4.36}$$

assuming prefect Doppler compensation on the carrier when down converting. The first term is the path loss where R_0 is the range to the target at pulse 0 and R_m is

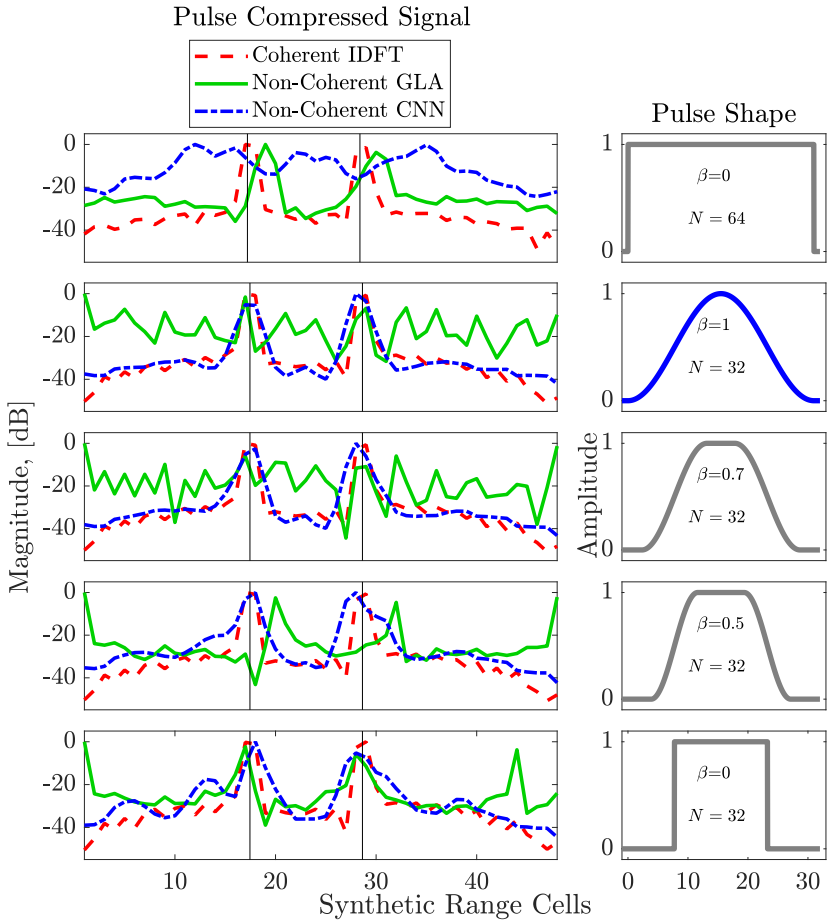


Figure 4.10: Comparison between the iterative GLA solution and the presented CNN solution for non-coherent pulse compression. For reference, the Coherent solution using the IDFT is also shown. The simulated scenarios show two single point scatterers "illuminated" with different pulse shapes in the form of raised cosine filters. The CNN was only trained on the raised cosine with shape factor $\beta = 1$ and N=32. The SNR was 25 dB. The vertical bars mark the true target positions. The number of range samples required for the GLA to work was $N_r = 9$ and the GLA estimate was therefore longer than the $N' = 48$ samples shown here.

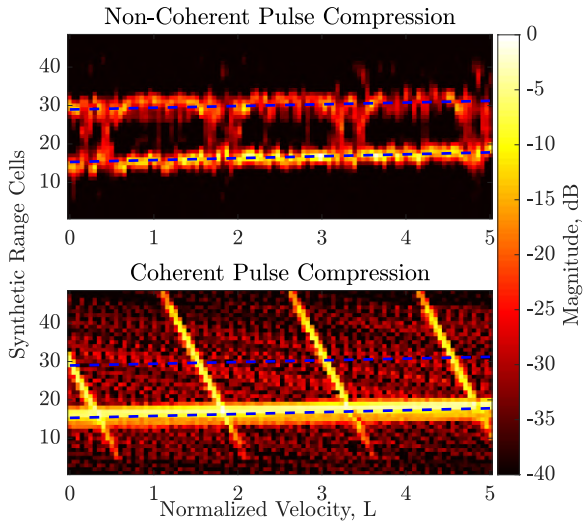


Figure 4.11: Scenario with two moving targets, where the top target moves with 0.9 times the velocity of the bottom target which is given on the x-axis. The coherent estimate is phase compensated for the velocity of the bottom target. The dashed lines show the true mean position of the targets during $M = 64$ pulses as a function of velocity. The y-axis shows the $N' = 48$ synthetic range cells. As an example, a normalized velocity of one is equivalent to 100 m/s with a 200 m pulse length and 1kHz PRF with N=32 and M=64. The mean SNR was 10 dB.

the range at pulse m. Using (4.6) the exponent in (4.36) can be written as

$$-j2\pi \left[\left(\frac{2\Delta\bar{r}L}{(M-1)(N-1)} - \frac{L}{2(M-1)} \right) m + \frac{Lm^2}{(M-1)^2} \right]. \tag{4.37}$$

The term L is a normalized velocity, i.e., the number of synthetic resolution cells the target has moved during the M pulses, not necessarily an integer. We see that this phase shift contains a linear term in m which will result in a spatial rotation of the estimated signal and a quadratic term in m which will result in a widening of the target.

Theoretically this can be compensated for by multiplying with the inverse phase, but this requires accurate knowledge of the relative velocity between target and radar. This is not a trivial problem, see [101], and [102]. The non-coherent solution is however unaffected by phase shifts. Fig. 4.11 shows such a scenario with two targets consisting of 50 point scatterers with exponentially distributed

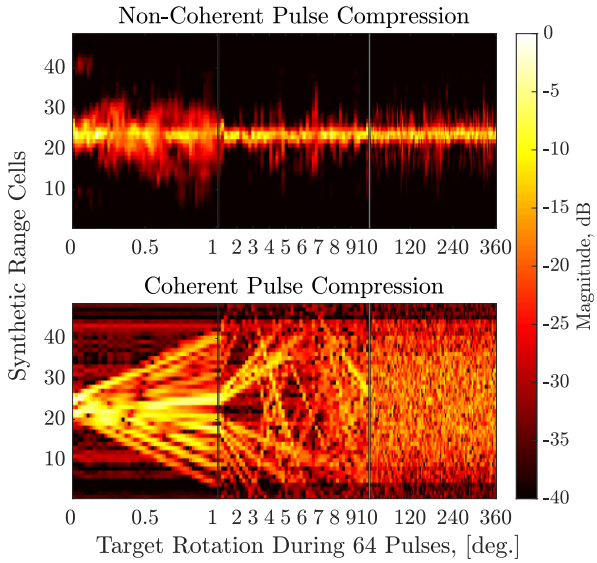


Figure 4.12: Scenario with a rotating target. The rotation in degrees during the M pulses is specified along the x-axis for three different intervals. The y-axis shows the $N' = 48$ synthetic range cells. As an example, a rotation of 1° with $M = 64$ pulses and 1kHz PRF corresponds to an angular velocity of 16° per second or 2.6 rpm. The mean SNR was 10 dB.

RCS, uniformly confined to five synthetic range cells. One target moves with 0.9 times the velocity of the other, specified on the x-axis. The coherent solution is only phase compensated for the bottom target, i.e. the velocity of that target is assumed known. Compared to the coherent estimate the non-coherent estimate is relatively unaffected despite not having been trained on such scenarios and does not require any knowledge of the relative velocity between target and radar.

An even more complicated case is when the target rotates, or equivalently, the aspect angle to the target changes during the M pulses when the target consists of more than one scatterer. Compensating for the phase and amplitude distortions caused by the slight change in radial distance to each scatterer require known relative positions and strengths of each point scatterer as well as the relative angular velocity. Fig. 4.12 shows a scenario with the same type of target as in Fig. 4.11. The compressed range profile is given as a function of the rotation of the target during the M pulses. Apart from a widening of the target in synthetic range, the non-coherent estimate shows little distortion when compared to the coherent estimate. The CNN has not been trained on this type of scenario.

4.4 Conclusions

In this study we have investigated the use of a CNN as a non-linear mapping from magnitude, or modulus, in discrete frequency domain to a corresponding magnitude in time domain. By so doing we have obtained an approximation of a high resolution range profile of targets with a non-coherent radar.

We found that for SNRs > 10 dB, the ability of the non-coherent CNN to resolve targets within a pulse length was a factor two worse than the equivalent coherent system, i.e. target spacing needed to be increased by factor two before targets peak-to-valley-ratio grew beyond 3 dB. The average peak-to-valley ratio with the CNN solution reached the corresponding value of the coherent peak-to-valley ratio after a target separation of 10 synthetic range cells, or 1/3 of the full pulse length.

The gain in SNR was estimated to be greater than 6 dB with 95% empirical probability for SNRs greater than 5 dB. Depending on the target it may be better in terms of detection probability and false alarm trade-off to integrate the required 64 pulses and perform detection on these rather than the CNN output. Provided one can buffer the 64 pulses, an appealing solution is to use pulse integration for detection of targets and later only apply the CNN solution on the batches with detected targets for locally increasing the resolution. This may decrease the processing time compared to evaluating the CNN on all batches.

We have only considered stationary scenarios and a stepped frequency waveform with a fixed interval between frequencies. The extension to dynamic scenarios and random frequency sampling is left for future work. We have however shown examples where the CNN yields reasonable results in dynamic scenarios despite not having been trained on these. This indicates that such CNN solutions could be of benefit even in coherent systems, as these are sensitive to phase changes which are unavoidable in dynamic scenarios. We have also not considered the effects of multipath propagation, interference, and other forms of noise which is left for future work.

All results in this study are based on simulated data and evaluation on real data is also left for future work. The required number of data samples for avoiding overfitting the network was found to be quite high for the chosen architecture, requiring an order of 10^5 samples. Obtaining this amount of real data for training may be difficult. We therefore believe that models of targets, such as the example presented in this study, will always be relevant for CNN based pulse compression, at least for transfer learning where the network is first trained on simulated data and then fine tuned on real data. We make no claim to have found the optimal architecture or training procedure for the CNN. Instead we have settled with showing that a rather simple CNN architecture can be used to gain decent results for non-coherent pulse compression. Solving this type of problem with more elaborate

network architectures such as the one presented in [94] is left for future work.

The results presented in this study offers new abilities to non-coherently process frequency agile radar signals for both imaging applications and detection. In case of detection, one can either add traditional methods such as a CFAR detector to the presented method, or look at an end to end approach where the network instead outputs a list of all targets with desired attributes using the same type of input. A study on this second approach is left for future work.

Chapter 5

Data Driven Track Before Detect

In this chapter, an ANN based method is used to detect weak target patterns in a batch of pulses, or slow time processing. This is done by first finding tracks. The tracks can be used to estimate good initial states for traditional trackers such as Kalman filters. The results and methods presented in this chapter are published in paper E; see page xi.

5.1 Introduction

Common to all radar systems is the trade-off between the detection probability and false alarm rate when detecting targets. An effective way to improve this trade-off is to first temporally process the received signal. If we assume that 1) the processing consists of temporal averaging, 2) all the integrated samples are independent and belong to the same category, either background noise/clutter or a target, and 3) the distribution of these categories are stationary during the integration interval, then the distribution of the averaged samples will asymptotically converge to Gaussian distributions with a variance that is scaled by the inverse of the number of averaged samples. Near perfect detection and false alarm rates can then be achieved if the mean values are different. There is however the problem of how to choose which samples in time to integrate over, and how many. This has given rise to various types of multi frame detection (MFD) [103] and track before detect (TBD) [104] algorithms. The focus in MFD is typically the detection of targets, while TBD methods will also produce tracks of the detected targets.

These methods typically build on some prior assumption of the target dynamics. The simplest is to assume that the velocity is such that the target is within the same resolution cell during the whole processing interval. In such cases we can simply average all cells corresponding to the same coordinate in range, angle and/or Doppler velocity. This assumption is only valid up to a certain integration interval. Another relatively simple assumption is constant velocity during the whole integration interval. We may then integrate the data over a set of straight hypothesis trajectories [105] using the Hough transform [106].

If however the considered segment contains several targets, and one of these is sufficiently strong, then all hypothesis tracks that cross the trajectory of the strong target will have enough energy to surpass the detection threshold, causing many false detections. It is therefore common to first threshold the data according to some CFAR scheme, and then integrate over the detections. This will of course come at a loss since the data is reduced to a binary resolution in magnitude.

Other TBD algorithms allow more complex trajectories and may use various methods such as dynamic programming [104], particle filters [107] and expectation maximization [108]. Common TBD methods are described and compared in [109]. Apart from computational complexity, some of these TBD algorithms will also suffer from the initialisation step, which is a common problem in tracking in general. A Kalman filter [110] will for instance follow a weak target quite well provided that the initial error of the target state is sufficiently small. As discussed in [111], most missed tracks are due to failed track initialisations. In [111], a neural network was used as a detector of potential targets in hybrid with a TBD algorithm, similar to the concepts in [112] and [113].

Data driven neural networks do not require models of the target and background signal distributions [113], nor models or of the target dynamics. Instead, data examples are required, and typically quite a lot of examples. The use of neural networks in target detection has been successfully demonstrated in various studies [114]. In particular, the use of neural networks in MFD has been evaluated in various approaches, either to remove false detections obtained by some traditional detection scheme [115], [116], to obtain detections that exploit the spatial and temporal correlations in the data [117–121], or to use neural networks to distinguish between true and false tracks [122].

In this study we introduce two new neural network based methods to be used within the TBD framework. The first is a CNN that can be seen as a noise reduction filter that exploits the spatio-temporal correlations of true trajectories. This has max pooling over channels which drastically reduces the number of computations, see Section 3.2.1. As it is only used for enhancing the target trajectories, it can be used as a prepossessing step to any existing TBD algorithm. The second network is a polynomial curve fitting MLP, used for non-causal association of detected targets

within a given window. Polynomial curves from different window segments are successively linked together, forming hypothesis tracks. Once, or rather if, these tracks contain energy that exceeds a set threshold, a detection is declared.

At this point the obtained tracks can be used as input to conventional tracking filters. If a Kalman filter is to be used, the detected track may be processed to estimate an appropriate dynamic model and initial state, as well as the variance of the process and measurement noise [110].

The presented method may be used in various radar applications, however we will focus on a surveillance scenario where the objective is to detect and track weak targets, e.g. drones, within small range segments. We evaluate our method on nonlinear target trajectories, multiple target tracking with crossing trajectories and with a mixture of strong and weak targets. We compare our results to a theoretically optimal detector rather than a specific type of TBD method.

The rest of this chapter is organized as follows. In Section 5.2 we present the data used in this study along with the assumptions made. In Section 5.3 we present the proposed method and the results are given in Section 5.4. Conclusions and directions for future work are given in Section 5.5.

5.2 Data Model

The input to our detector is a range-time (RT) frame $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_{N_T}] \in \mathbb{R}^{N_R \times N_T}$ containing the power from a range segment with N_R range cells in N_T batches. Each batch may be formed from a number of pulses, and the processing within a batch may or may not be coherent. We do however not assume any coherence between batches. As stated in Section 5.1, the proposed data driven method does not require any particular model of the target dynamics, nor the power distributions of targets and background. For data generation purposes however, we still need these models. The ones chosen in this study could be replaced with other models or actual measurements. The chosen models for the data generation are as follows. The number of targets within each frame, M, is randomly selected. The trajectory of each target is generated from a Gaussian process with length N_T. Each element of the covariance matrix used for this is given by the Gaussian kernel

$$\boldsymbol{\Sigma}[i, j] = \sigma^2 e^{-((i-1)-(j-1))^2/l^2} \tag{5.1}$$

for $i = 1, ..., N_T$ and $j = 1, ..., N_T$. The parameter σ^2 was constant while the length scale parameter l^2 varied. Some examples of target trajectories for different intervals of l^2 are shown in Fig. 5.1. All generated tracks were initially normalized to have zero mean before adding a bias that was uniformly sampled between 0 and the length of the considered range segment ΔR.

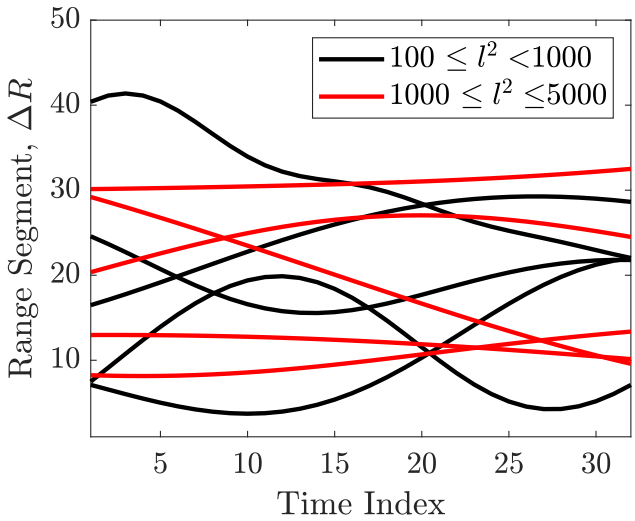


Figure 5.1: Examples of trajectories for different intervals of the length scale parameter l^2. In all cases, the parameter $\sigma^2 = 50$.

Equivalent to what would be obtained with a FMCW radar, each batch $\boldsymbol{x}_t$ in $\boldsymbol{X}$ is generated as a sum of sinc functions

$$\boldsymbol{x}_t = \left| \sum_{m=1}^{M} \sqrt{s_{m,t}} \text{sinc} \left((\boldsymbol{r} - 1) \frac{\Delta R}{N_R - 1} - r_{m,t} \right) + \boldsymbol{w}_t \right|^2 \tag{5.2}$$

for range cells $\boldsymbol{r} = [1, ..., N_R]^T$ where $r_{m,t}$ is the range position of target m at batch t and $\boldsymbol{w}_t \in \mathbb{C}^{N_R \times 1}$ is a vector with zero mean uncorrelated complex Gaussian noise with variance σ_w^2. $s_{m,t}$ is the instantaneous gain of target m at time t. This is drawn from a Chi-square distribution with four degrees of freedom and mean $\bar{s}_m$, corresponding to Swerling III and IV targets [30] depending on the time between batches. The SNR of each target is then $\gamma_m = \bar{s}_m / \sigma_w^2$.

Note that we have not included the path loss proportional to R^4. This simplifies the evaluations since a target will then have the same SNR throughout the entire track, regardless of range. The simplification is approximately true if the range ΔR is small relative to the range to the first range cell.

We normalize $\boldsymbol{X}$ by dividing by the median of all elements in $\boldsymbol{X}$. Depending on how many targets a sample of $\boldsymbol{X}$ has, the median may vary somewhat. This variation will have an effect on the detection probability as we will see in

Table 5.1: Parameters for Data Generation

Parameter	**Value**
σ^2	50
l^2	[log(100), log(5000)], uniformly sampled in log scale
ΔR	50
N_R	64
N_T	32
M	[1 10], uniformly sampled integers
γ	[0, 20] dB, uniformly sampled in log scale
N_{tr}^1	190000
N_{va}^1	10000
N_{te}^1	$(21 \times 10000)/M$
N_{tr}^2	380000
N_{va}^2	20000

Section 5.4.

All parameters used for generating data are shown in Table 5.1. Note that the parameters have no unit. The number of training and validation samples when training the CNN is N_{tr}^1 and N_{va}^1, respectively. For the test samples, we generate 10000 tracks for SNR values 0,1,2,...,20 dB. The number of tracks in each test frame varied depending on the scenario, and the number of test samples N_{te}^1 thus varied accordingly.

5.3 Detection Method

The RT-frame is first filtered with the CNN shown in Table 5.2. The first four layers have maxout [123] or "max-channel" activations, where the largest value in each channel is selected. These layers can thus be regarded as a bank of F filters where the filter that is best matched to a particular 9×9 segment is selected. By only passing the maximal value, the output size of each such layer is always $N_R \times N_T$, reducing the number of parameters in the next layer by a factor F. The final output has linear activation and the label is a binary matrix $\boldsymbol{Y} \in \mathbb{R}^{N_R \times N_T}$ with value one if a cell contains a target and zero otherwise. The network was trained with the truncated empirical MSE loss

$$L = \frac{1}{N_b N_R N_T} \sum_{n=1}^{N_b} \sum_{r=1}^{N_R} \sum_{t=1}^{N_T} \left(\hat{\boldsymbol{Y}}'_n[r,t] - \boldsymbol{Y}_n[r,t] \right)^2 \tag{5.3}$$

Table 5.2: Neural Network Architecture

CNN, Input: $\boldsymbol{X}$, output: $\hat{\boldsymbol{Y}}$				
Layer type	Weight Shape	Zero Padding	Output Shape	Activation
Convolutional	F: 10, K: 9×9	[4,4]	64×32	Max-Channel
Convolutional	F: 10, K: 9×9	[4,4]	64×32	Max-Channel
Convolutional	F: 10, K: 9×9	[4,4]	64×32	Max-Channel
Convolutional	F: 10, K: 9×9	[4,4]	64×32	Max-Channel
Convolutional	F: 5, K: 9×21	[4,10]	64×32×5	ReLu
Convolutional	F: 1, K: 9×21	[4,10]	64×32	Linear
MLP[1], Input: $\boldsymbol{Z}$, output: $\hat{\boldsymbol{p}}$				
Layer type	Weight Shape	Zero Padding	Output Shape	Activation
Fully Connected	16×121	-	16×1	ReLu
Fully Connected	16×16	-	16×1	ReLu
Fully Connected	16×16	-	16×1	ReLu
Fully Connected	16×16	-	16×1	ReLu
Fully Connected	16×3	-	3×1	Linear
MLP[2], Input: $\boldsymbol{Z}$, output: I				
Layer type	Weight Shape	Zero Padding	Output Shape	Activation
Fully Connected	16×121	-	16×1	ReLu
Fully Connected	16×16	-	16×1	ReLu
Fully Connected	16×1	-	1×1	Sigmoid

F: Number of filters/kernels. K: Kernel size. Total number of parameters in CNN: 5176, MLP[1]: 2819, MLP[2]: 2241

where

$$\hat{\boldsymbol{Y}}'_n[r,t] = \min(1, \max(0, \hat{\boldsymbol{Y}}_n[r,t])) \tag{5.4}$$

and $N_b = 128$ is a batch size used in stochastic gradient descent. By truncating the CNN output $\hat{\boldsymbol{Y}}$ in (5.4), we allow targets with high power to surpass the label value of one without increasing the loss. After the filtering, all elements of $\hat{\boldsymbol{Y}}$ lower than a threshold T_0 are set to zero. Consistent with the loss function we set $T_0 = 0$.

Next we associate elements in $\hat{\boldsymbol{Y}}$ to form tracks. For each cell in $\hat{\boldsymbol{Y}}$ that exceeds a threshold T_1, a window of ± 5 cells in both range and time is selected. We refer to such cells as a CUT, and we denote the selected segment as $\boldsymbol{Z} \in \mathbb{R}^{11\times 11}$. This is zero padded if the CUT is close to an edge. By only processing cells that exceed T_1 we can remove much of the computational load. Assume that the CUT is part of a true track, and let $\tilde{\boldsymbol{r}} \in \mathbb{R}^{11\times 1}$ and $\tilde{\boldsymbol{t}} = [-5, -4, .., 4, 5]^T$ constitute a segment of this track that is within the window and centered at the CUT. Here $\tilde{\boldsymbol{r}} \in [-5, 5]$ signifies

the range values and $\tilde{\boldsymbol{t}}$ the corresponding time values. We model this segment as $\tilde{\boldsymbol{r}} = \boldsymbol{A}\boldsymbol{p}$ where $\boldsymbol{p} \in \mathbb{R}^{k\times 1}$ are polynomial coefficients and

$$\boldsymbol{A} = \left[\tilde{\boldsymbol{t}}^{k-1}, ..., \tilde{\boldsymbol{t}}^{0}\right]. \tag{5.5}$$

The objective of MLP^1 and MLP^2 in Table 5.2 is then to estimate $\boldsymbol{p}$ from $\boldsymbol{Z}$.

MLP^1 is used to estimate $\boldsymbol{p}$ in a data driven approach. If however $\boldsymbol{Z}$ only contains one track, this estimate can instead be obtained using least squares (LS). Let $\hat{\boldsymbol{r}} \in \mathbb{R}^{11\times 1}$ be a vector with the argument of the maxima subtracted by 6, of each batch in $\boldsymbol{Z}$. Let $\boldsymbol{v} \in \mathbb{R}^{11\times 1}$ be the corresponding maximum values. We then minimize $||\boldsymbol{V}(\hat{\boldsymbol{r}} - \boldsymbol{A}\hat{\boldsymbol{p}})||_2^2$ where $\boldsymbol{V} = \text{diag}(\boldsymbol{v})$. The weighted LS solution to this is

$$\hat{\boldsymbol{p}}^{\text{LS}} = (\boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{V}\hat{\boldsymbol{r}}. \tag{5.6}$$

We use the weighted LS solutions since the maximum value in a batch may in some cases be zero due to T_0. This solution will fail when $\boldsymbol{Z}$ contains several tracks, in which case we should use the estimate from MLP^1 instead, $\hat{\boldsymbol{p}}^{\text{MLP}^1}$. To accomplish this we weight these as

$$\hat{\boldsymbol{p}} = I\hat{\boldsymbol{p}}^{\text{MLP}^1} + (1 - I)\hat{\boldsymbol{p}}^{\text{LS}} \tag{5.7}$$

where $I \in [0, 1]$ is the scalar output of MLP^2 and acts as an indicator of which estimate to use. MLP^1 and MLP^2 were trained simultaneously using (5.7) in the empirical MSE loss function with the true values of $\boldsymbol{p}$ as label, see Section 3.3.1. For this we used N_{tr}^2 and N_{va}^2 training and validation segments from $\hat{\boldsymbol{Y}}$, see Table 5.1. In all these samples the CUT was above T_1 and the corresponding value in $\boldsymbol{Y}$ was one. Since a value of $I = 0.5$ suggests that both the LS and MLP^1 solutions are equally good, we scale I as

$$I' = 2(\max(0.5, I) - 0.5) \tag{5.8}$$

in the final evaluations, where we replace I in (5.7) with I' in (5.8). We thus give preference to the LS solution. As described in Section 3.3, all networks were trained using stochastic gradient descent with Adam optimisation.

The degree of the polynomial curve should match the observation time and target dynamics. In the considered case we found that a second order curve fit was sufficient, i.e. $\hat{\boldsymbol{p}} \in \mathbb{R}^{3\times 1}$. The polynomial curves are then linked to form hypothesis tracks as follows. Let $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_{N_T}] \in \mathbb{R}^{N_R\times N_T}$ be a zero valued matrix. Starting at the cell in $\hat{\boldsymbol{Y}}$ with the highest value that also exceeds T_1, we estimate $\boldsymbol{p}$ around this CUT using MLP^1 and MLP^2. All cells in $\hat{\boldsymbol{Y}}$ crossed by the curve $\lfloor\boldsymbol{A}\hat{\boldsymbol{p}}\rceil$ are then added to $\boldsymbol{B}$ around the CUT. Here $\lfloor\cdot\rceil$ signifies rounding to

the nearest integer. Since a target may lie between two cells, we also add the two neighboring range cells to $\boldsymbol{B}$ for each batch. We repeat this for all samples within the current segment $\boldsymbol{Z}$ that exceed T_1, i.e. a new curve is drawn for each such cell. If these curves overlap the current hypothesis track in $\boldsymbol{B}$ by at least two cells, the corresponding values in $\hat{\boldsymbol{Y}}$ are added to $\boldsymbol{B}$. The hypothesis track has range index $r_t^{\max} = \arg\max \boldsymbol{b}_t$ at batch $t \in \mathcal{T}$. $\mathcal{T}$ is the set of time indexes in the hypothesis track. These are such that there is at least one non-zero value in $\boldsymbol{b}_t$. We then sum samples along the hypothesis track in the input $\boldsymbol{X}$

$$u = \frac{1}{N_T} \sum_{t \in \mathcal{T}} \boldsymbol{X}[r_t^{\max}, t]. \tag{5.9}$$

If the hypothesis track is at least T_L batches long, and if the score u exceeds a threshold T_2, a detection is declared, and all cells in the hypothesis track cannot be used in another track.

If these criteria are not met, the current CUT is moved along the hypothesis track by one batch, and the process is repeated for the new CUT. The CUT is moved to the right or left depending on which increases the score in (5.9) the most. We terminate the track expansion if the score exceeds T_2. Otherwise the tracking is terminated when the first and last batch are reached, or if the window at an edge of the hypothesis track only covers zero valued cells. If the score is less than T_2 at this point, the hypotheses track is rejected and no detection declared. Whether the track is declared as detection or not, all values in $\hat{\boldsymbol{Y}}$ along the hypothesis track are set to zero, including the two nearest range cells in each batch.

A new track is then initialised at the current highest value in $\hat{\boldsymbol{Y}}$ that also exceeds T_1 and has not yet been assigned to any hypothesis track (detected or rejected), and so forth until all such cells in $\hat{\boldsymbol{Y}}$ have been processed. Note that if a target is strong a track may be detected at the first CUT. This is desirable since longer tracks have an increased risk of containing cells from different tracks due to incorrect associations. It may however also result in several detections per target. This is however not considered as a problem in this study.

5.4 Results

We begin our evaluations by looking at the detection probability of individual cells before and after the CNN filter. Fig. 5.2 shows the ROC curves for different SNR values. We ignore cells adjacent to true tracks. The false detection count is thereby the total number of detected cells that are not true targets nor adjacent to a target. We note a significant improvement when applying the CNN filter.

Next we consider the TBD detection probability. To simplify our results, we

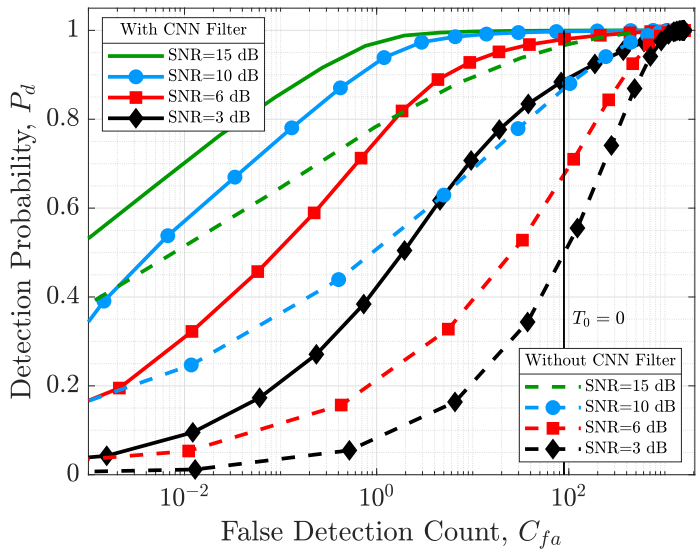


Figure 5.2: Empirical ROC curves for different SNR values. Dashed lines are without the CNN filter, and solid lines are with the CNN filter. The threshold T_0 only applies to the curves with the CNN filter.

limit our evaluations to a case where the track initialisation threshold is set to $T_1 = 0.5$, the track detection threshold is $T_2 = 3$ and track length threshold is $T_L = 7$. These values are based on empirical evaluations and showed good ROC trade-offs for this particular dataset. We benchmark the results to a theoretically optimal, but practically unfeasible, detector were we assume that all possible target trajectories in the test set are known. We refer to this as the ideal detector.

With 21 SNR values between 0 and 20 dB, and 10000 samples for each SNR, we have 210000 tracks. About 150000 of these are unique when quantized to the N_R resolution cells in range and normalized such that all tracks begin at the same range cell. To cover all possible cases, all these unique tracks are shifted to each of the N_R range cells. We discard any case where a track goes outside the considered range segment with the motivation that these would be covered had we chosen a different segment. This results in about eight million hypothesis tracks to test for each RT-frame making it an expensive solution from a computational perspective. This solution is however optimal in the sense that we are guaranteed to test the energy in all true tracks, and if detected, the MSE in range cells will be zero.

For the proposed solution, we consider a detected track as a true detection if

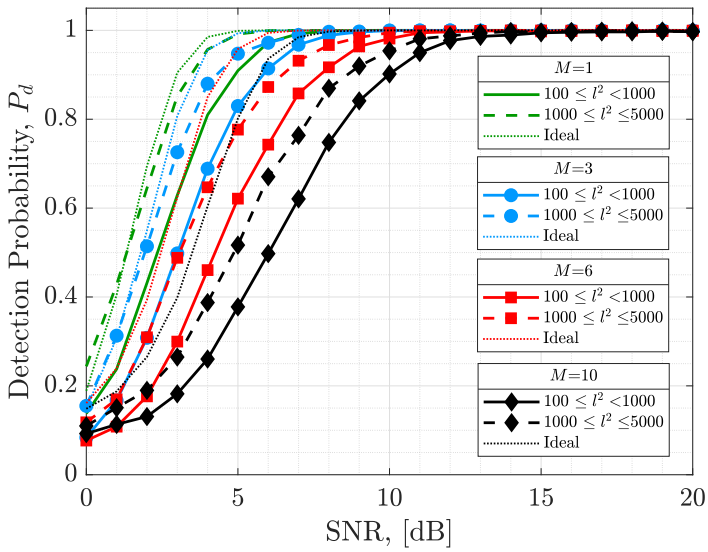


Figure 5.3: Empirical detection probability as a function of SNR for four different cases. The threshold for detection was $T_2 = 3$ and $T_L = 7$. For $M = 1$ and low SNRs, the proposed solution appears to surpass the ideal benchmark solution. The reason for this is that we only tested the true target tracks in the ideal case. At low SNRs, hypothesis tracks that partially overlap with true trajectories may have higher score than the actual track leading to a slightly increased detection rate but increased MSE.

the first CUT is adjacent to, or on a true target track. The empirical detection probability as a function of SNR is shown in Fig. 5.3 for scenarios with different number of targets M. The benchmark detector is shown as the ideal detection probability. We see that for $M = 1$ the proposed solution comes close to the ideal detection probability, especially when $100 \leq l^2 \leq 1000$.

The P_d decreases with higher M for two reasons. The first is that we divide the RT-frame by its median. More targets in the frame will slightly increase the median value, leading to a slightly reduced score value for each track. Note that the threshold T_2 is the same regardless of M. The second reason is that the number of trajectory crossings increase with M. Since we only allow a cell to belong to one track, cells at trajectory crossings will only add to the score of at most one trajectory, thus reducing the score of the other. Note that we normalize the score by N_T, regardless of the length of the hypothesis track. These crossings may also cause true tracks to become segmented into smaller and separate hypothesis tracks,

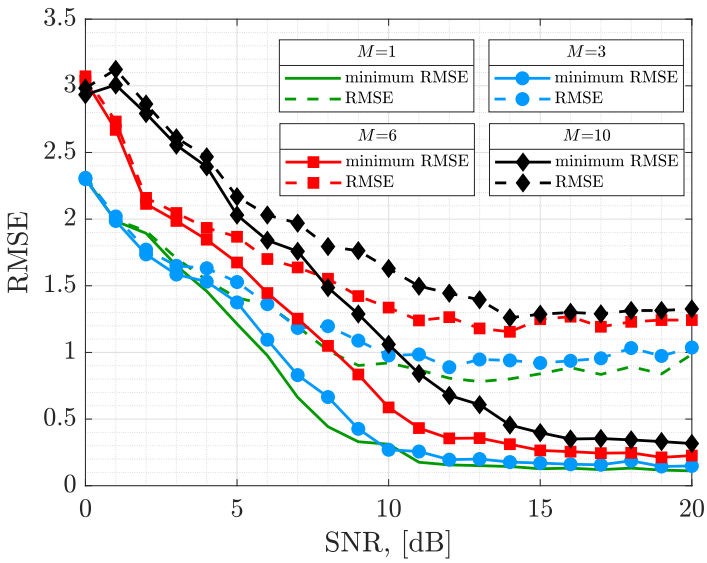


Figure 5.4: The RMSE in range cells. A target may be detected in several segments. The dashed lines mark the RMSE when taking the mean error of all detections per track, and the solid line marks the RMSE when only considering the segments with the minimum error in each track. At least one segment/track per detected target will follow these curves.

where none surpass T_2. Such segmentation into separate tracks does however not apply to the ideal case.

Since we declare a hypothesis track as a detection as soon as the score is sufficiently high, a strong target may be detected as several track segments, and the squared error may thus vary depending on segment. Let the minimum root mean square error (RMSE) be the RMSE of the best segment of each detected track and let the RMSE be the mean of all segments for all targets. These are shown in Fig. 5.4. The error is in range cells, and the 3 dB range resolution of the radar is from (5.2) about 1.12 range cells.

The false alarm count in tracks, $\tilde{C}_{fa}$, is shown in Table 5.3 for the presented solution and the ideal benchmark case. Similar to the detection probability, the number of false tracks depend on the number of true tracks, and for the same reasons. As we only have white Gaussian noise in the considered scenarios, the score of a false hypothesis track will be the sum of exponentially distributed random variables. Since we normalize by the median, the mean of these numbers will be

Table 5.3: Empirical False Track Count for $T_2 = 3$

M	0	1	3	6	10
$\tilde{C}_{fa}$, Proposed Detector	0.04	0.025	0.01	0.004	0.001
$\tilde{C}_{fa}$, Ideal Detector	0.6	0.4	0.4	0.35	0.1

$1/\log(2)$, when $M = 0$. The score will then follow an Erlang distribution with false alarm probability

$$p_{fa} = \sum_{n=0}^{H-1} \frac{1}{n!} e^{-\lambda_T T_2} (\lambda_T T_2)^n \tag{5.10}$$

where H is the cardinality of the hypothesis track and $\lambda_T = N_T \log(2)$. Setting $T_2 = 3$ and assuming $H = N_T = 32$ gives $p_{fa} \approx 10^{-6}$. With eight million hypothesis tracks, the false alarm count would then be about eight, had all tracks been independent. Due to significant overlap between many of the tracks, the actual false alarm count for the ideal case is 0.6 tracks per RT-frames when $M = 0$. We note that the false alarm count for the proposed solution is considerably lower than the ideal case at the expense of reduced detection probability at low SNRs and increased MSE.

Finally we show some examples. Fig. 5.5 shows a case with four targets with 5,4,3 and 2 dB SNR. All four targets are detected. Fig. 5.5 also shows examples of the polynomial curve fit in $\boldsymbol{Z}$ for two different segments. In Fig. 5.6 we show a case with a large target that spans more than one range cell [124]. Although the proposed solution detects this as a single track, we see tendencies to two parallel tracks in image B in Fig. 5.6. The second "track" is in this case always lower than T_1. A longer target would lead to several parallel tracks with the current solution. Note that the CNN was not trained on such wide targets. We also note in Fig. 5.6 that target 3, a relatively weak target in SNR, is accurately tracked despite it crossing the trajectory of target 2, a relatively strong target in SNR. The two stronger targets are detected in multiple segments whereas target 3 required almost all batches in order to accumulate a sufficiently high score for detection.

Last we show an example from an actual radar measurement. Fig. 5.7 shows an RT-frame obtained with the FMCW radar in Section 2.9. The measurement is on a small drone. We note from image A in Fig. 5.7 that the true trajectory, accurately detected in image C, is difficult to distinguish even for the human eye. The drone trajectory is detected as track 2 in Fig. 5.7, where the drone begins to fly towards the radar. In Fig. 5.8 we show the same example but with a larger RT-frame. In this case we convolved the CNN and MLP filters over the entire input and merge the resulting tracks. Although we see some false detections in image B in Fig. 5.8, most

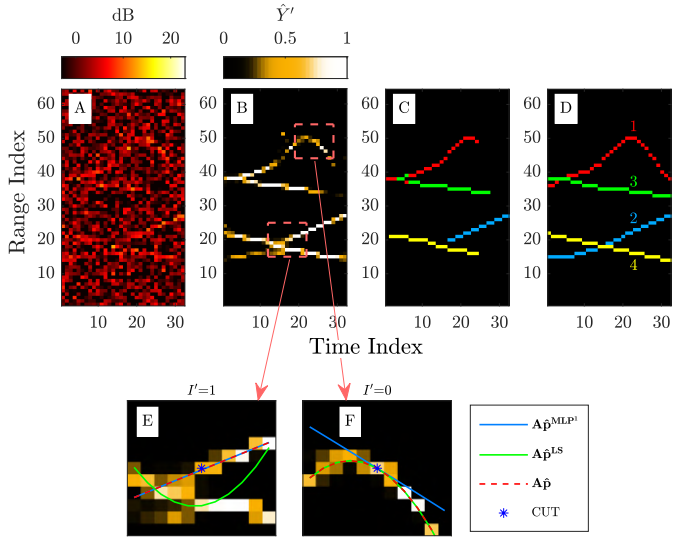


Figure 5.5: RT-frame example with weak targets. The true tracks are shown in image D. The SNRs of targets 1-4 are 5, 4, 3, and 2 dB, respectively. Image A shows the Input RT-frame $\boldsymbol{X}$ and image B shows the RT-frame after the CNN filter, $\hat{\boldsymbol{Y}}'$. Image C shows all tracks that surpassed the threshold T_2. We see that all four tracks are detected. The RMSE for the segments in image C was 0.85 for target 1, 0.35 for target 2, 0.46 for target 3 and 0.42 for target 4. Images E and F show examples of the association step in the 11×11 segments $\boldsymbol{Z}$. In E, the MLP^1 solution is used since $I'=1$ and the LS solution is used in F since $I'=0$.

detected tracks appear consistent with the barley observable drone path in image A in Fig. 5.8. The false tracks can be filtered out by considering overlaps of detected tracks from consecutive but overlapping segments. We see an example of this in image C in Fig. 5.8 where almost all false tracks appear to be removed. We also show examples of detections using a conventional OS-CFAR detector. The window size (in range) was ±32 samples and the rank order was 38 (60% of window). We show examples of the detections obtained with four different threshold levels in D-H in Fig. 5.8. Note that unlike the detected tracks in C and B, the OS-CFAR detections are not tracks, i.e. there is no association between two detections. We see that a high threshold is required to obtain a low false alarm rate, at which point very little of the drone path is detected.

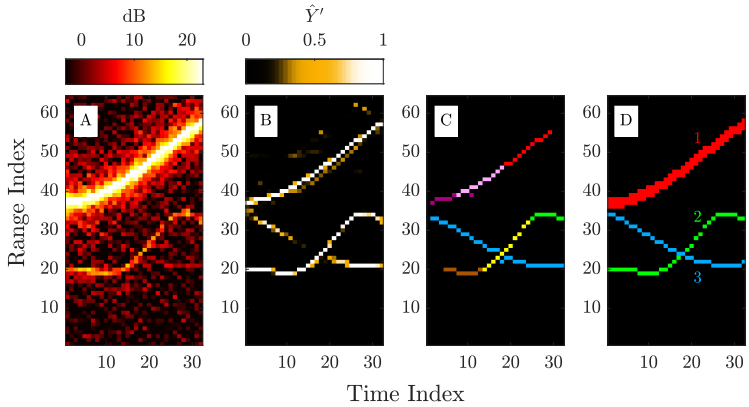


Figure 5.6: RT-frame example with strong and weak targets. Image D shows the true tracks of targets 1, 2 and 3 with SNRs 25, 15 and 4 dB, respectively. Image A shows the Input RT-frame $\boldsymbol{X}$, and image B shows $\hat{\boldsymbol{Y}}'$. The detected trajectories are shown in image C. The RMSE for target 1 was 0.41 and the minimum RMSE was 0.33 (red colored segment to far right in C). For target 2 the RMSE was 0 for all three segments and The RMSE of Target 3 was 0.54.

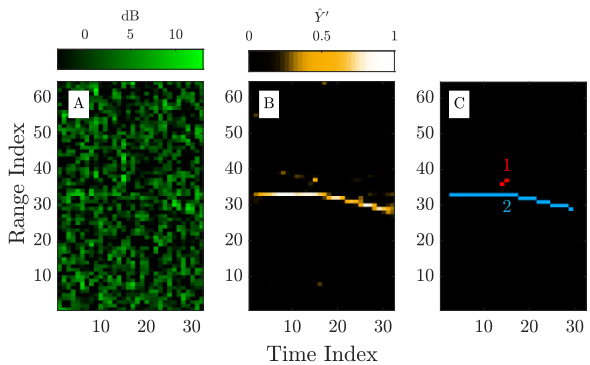


Figure 5.7: Example of a real radar measurement obtained with the FMCW radar in Section 2.9. The RT-segment in image A covers 132-192 m in range with one meter resolution. Each time index is a single FM sweep at constant angle. The time between two indices is 100 ms. The measurement is of a DJI Mavic 2 Pro, a small quadcopter drone. Image B shows the CNN output. At time index 17, the drone starts moving towards the radar. Two tracks were initiated, track 1 and 2 shown in image C. These had scores 0.46 and 3.3 respectively, i.e. only track 2 accumulated enough energy to become a detection.

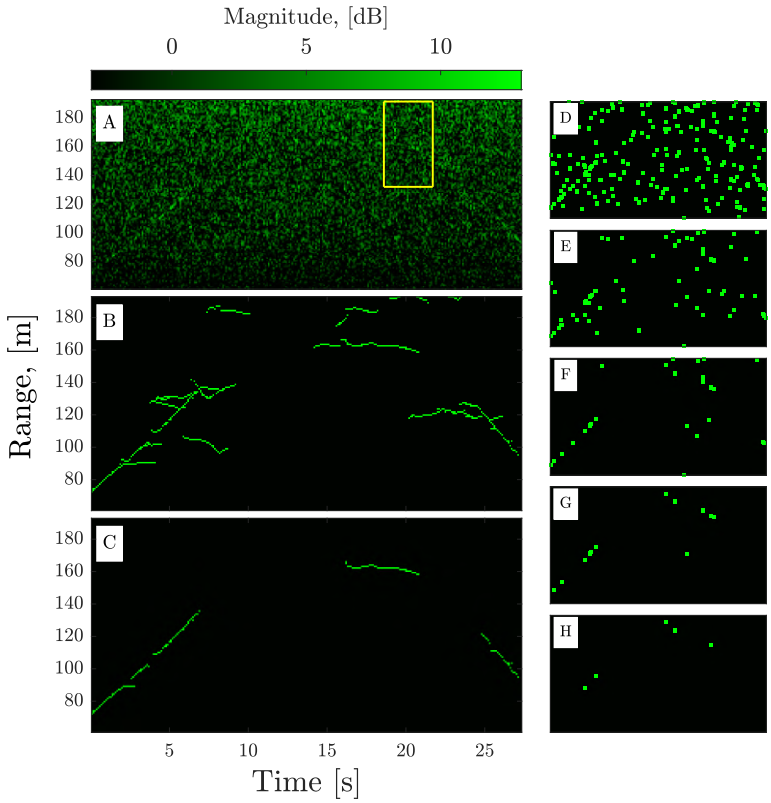


Figure 5.8: Example with a larger RT-frame. Image A shows an RT-frame obtained with the FMCW radar described in Section 2.9. The measurement is of a DJI Mavic 2 Pro, a small quadcopter drone. Beginning at approximately 70 m in range, the drone flies away and reaches a range of about 160 m after 12 seconds. At 20 seconds it begins to fly back towards the radar. The entire frame is processed in RT segments of 64×32, the input size to the CNN. This is shown as the yellow segment in image A, which is also the example shown in Fig. 5.7. Image B shows the logical sum of detected tracks using the CNN and MLP filters. These were convolved over the full RT-frame in A with stride [8,4] in range and time. The true trajectory appears to be partially outlined in image B, together with some false alarms. Most of these false alarms could be filtered out by considering the overlap of detected tracks from consecutive segments. An example of this is shown in image C we we have only kept tracks that overlap eight other tracks with at least ten cells (except around the edges where both these thresholds are reduced to 4). For reference, OS-CFAR detections are shown in image D-H. The CFAR threshold levels were 6, 8, 10, 12 and 14 in D, E, F, G and F respectively.

5.5 Conclusions

In this study we have presented a data driven TBD method based on ANNs. For certain scenarios, the empirical detection probability was close to an optimal system that tests all possible tracks. The required SNR for 99% detection probability with Swerling III/IV targets was between 5 and 13 dB depending on scenario. The false alarm count was low when compared to the benchmark detector, at the cost of increased RMSE and missed detections at low SNRs. The presented solution was trained on simulated data. This could be replaced with other data models or actual data. The method was evaluated on a limited set of real FMCW measurements and appears to work well despite having been trained on simulated data only. Further tests on this is left for future work as well as extensions to higher dimensional inputs such as range-Doppler-angle-time “frames”.

Chapter 6

Drone Classification

In this study we evaluate a CNN classifier's ability to discriminate drones from non-drone targets, such as birds, when these are not represented in the training data. We present two different methods for improving the classification accuracy on samples that are outside the training distribution. In the first, we add a synthetic data class to the classifier. This class is intended to represent samples from unknown drones. In the second method, we only train the classifier on non-drone data, and treat anomalous samples as drones. This method is also aided by synthetic drone data. The results and methods presented in this chapter are published in paper B and D; see page xi.

6.1 Introduction

A drone, or unmanned aerial vehicle (UAV), can be used to deliver harmful payloads, cause disturbances and/or collect footage of sensitive sites. The ability to accurately detect and classify drones is therefore of great importance in surveillance systems.

Although the same can be said about the larger UAV types, mainly used by military, this study considers small propeller driven drones often with a diagonal wheelbase of less than one meter. Over recent years these types of drones have become easier to acquire and handle. As a consequence, an increased concern over such drones potential to be used as weapons has motivated much research in drone classification methods. Apart from radar, this type of classification may be based on visual camera sensors, radio frequency based methods and acoustic methods. See [125] for a comprehensive overview of these.

In the literature, the non-drone class is mainly constituted of birds, but depending

on what area is under surveillance humans, animals, etc., may also become relevant non-drone targets. Airplanes and other aerial and ground vehicles can often be discarded as false alarms by their RCS values alone. Birds, humans and drones however may very well have comparable RCS as shown in Section 6.2.1.

Radar based solutions for drone classification can be divided into one of at least two basic approaches. The first is trajectory based where target attributes, such as position, velocity, and RCS, are analyzed over time to determine whether an object is a drone or not; see e.g. [126]. This is suitable for long range detection as it only needs a SNR that is high enough for tracking. It does however require relatively long measurement times, such as 60 s in [127], in order to obtain trajectories with meaningful features. At short ranges the decision process must be much quicker. This requirement is met in the second main approach which is based on the radar micro-Doppler spectrum created by the motion of the propellers. These require a measurement time ranging from the order of seconds, [128, 129], to as low as 50 ms [130]. The propeller RCS can be several orders of magnitude below the drone's total RCS; e.g. see [131, Fig. 2]. A high SNR and/or integration time is therefore required for the micro-Doppler features to become visible, making long range detection more difficult.

The rotation of a drone's propellers will give rise to micro-Doppler components that are different when compared to the micro-Doppler components that arise from the flapping of a bird's wings [130]. Pattern recognition techniques such as ANNs are therefore frequently used as classifiers of radar micro-Doppler. E.g. see [125, 130, 132–134]. The results reported from such studies are often impressive. However, as is the case with any method, they are not without problems. In [135] and [136] neural networks are described as "black box" methods since it is not always clear what the classification decision is based on, or how the classifier will behave on data outside the distribution of the training set. This is an important problem since it is difficult to collect data on all present and future types of drones. Birds are perhaps less cooperative in the measurement process, but are at least freely available and are not likely to change much in the foreseeable future.

We address this problem with two different approaches using CNNs. The considered signal processing chain is illustrated in Fig. 6.1 and is the same for both methods. We assume that targets have been detected within a scan, e.g. by some conventional CFAR method, and that small segments surrounding the detection are available for the target recognition steps. In the first approach, we train a CNN classifier on both synthetic and real data in order to make the classifier more robust to out-of-distribution drones. We then evaluate the CNN on drone types not present in the training data. To make the classifier a bit more transparent, we also look at what parts in the Doppler spectrum receive heavier weights in the classification step. In the second approach, we train a CNN to recognise normal non-drone

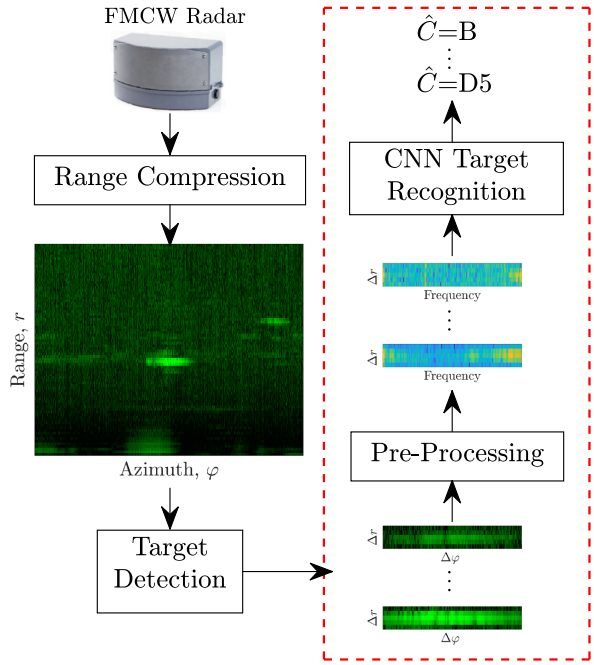


Figure 6.1: Illustration of the considered signal processing chain. Scan segments, centered over targets, are preprocessed and fed to the classifier. The two targets in the radar image are D5 (cf. Table 6.1) and a small bird, possibly a fieldfare. Both were accurately identified with the CNN presented in Section 6.3.2. This study addresses the recognition step, marked by the red dashed lines.

classes. Any "outliers" or anomalous samples are classified as potential drone targets.

Anomaly detection in high dimensional data, such as the micro-Doppler spectra considered here, is challenging [137, 138]. A fundamental problem is that the Euclidean distance, an effective metric for anomalies in low dimensional space, has little to no meaning in high dimensional space [139]. A common remedy is therefore to first project the data to a lower dimension, e.g. by using autoencoders [140]. This is unfortunately not a straightforward solution since there are no guarantees that the information lost in the encoding will not be of significance. For instance, if linear PCA is used, the encoder projection matrix is determined by the eigenvectors of the covariance matrix of the training data. There are no guarantees that the anomalous classes will be projected differently onto these than

the normal classes.

Our approach is to generate data from a mathematical model that is assumed to span the full "spectrum" of micro-Doppler spectra. We then train a CNN based autoencoder to project this model data to a lower dimension. By using a model instead of the data from the known classes, we aim to better encode differences that may appear between different classes, as well as reduce the required number of real samples. The encoder is then used to project real samples from a known non-Drone class. Finally we form a multivariate Gaussian model from the encoded samples and define anomalies using the log likelihood as well as assumptions derived from the model. With this second approach the classifier does not use a single sample from real drones in the training process.

The sensor that we use is the mechanically scanning FMCW sensor presented in Section 2.9. The dwell time within the 3 dB beamwidth is limited to 4 ms. We will however use processing intervals of 9 ms in this study. This is still a relatively short processing time for this type of classification. Performing the classification on a scan basis is also quite rare. Some examples are [135] and [141]. Although locking on to a target may yield higher accuracies due to longer dwell time, it may still be desirable to obtain an initial classification on a scan basis. The radar would thereby not need to lock on to every potential target for further analysis.

In [135], some 20 handcrafted features were used for non-neural network based classification, and single-scan accuracies of e.g. 75% for birds and a total of 94% for at least five different drones were reported. The required SNR for achieving these results were however not stated. As mentioned, a high SNR is often required for micro-Doppler features to be distinguishable from spectrum noise. According to [142] many of the impressive results from CNN classifiers are conditioned on very high SNRs, and significant drops in accuracy were reported already at 20 dB SNR when adding simulated noise in their datasets. We will address this by also looking at the SNR dependence of our results.

6.2 Data and Preprocessing

In this section we present the data obtained with the FMCW sensor and the synthetic data model.

6.2.1 Real Data

Data from a total of six different drones together humans (H), birds (B) and trihedral corner reflectors (R) were collected with the radar in Section 2.9. The dataset is published in [143]. The reflector class was introduced to represent targets with no micro-Doppler. All other classes are measurements of targets in motion. A

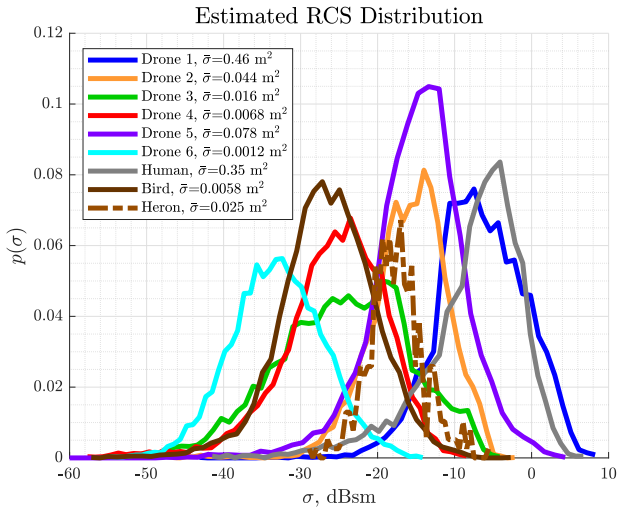


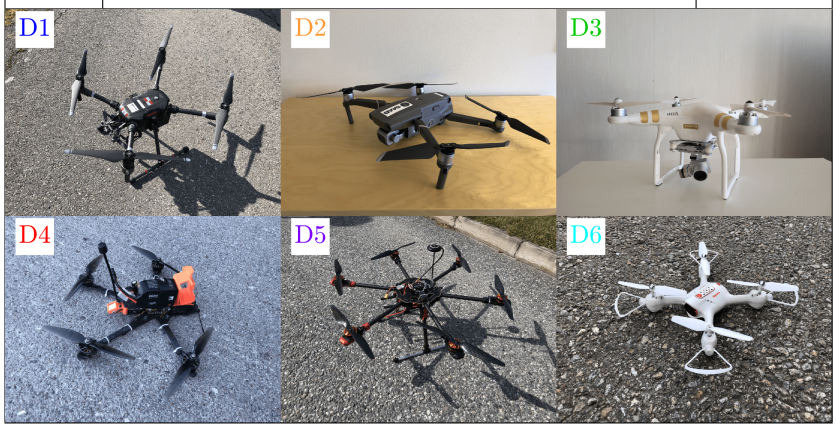
Figure 6.2: Distributions of estimated RCS values at 77 GHz. The distributions are approximately exponential. The heron is a significantly larger bird than the other species in the bird class and is therefore plotted separately. The mean RCS for each class is shown as $\bar{\sigma}$.

perfectly still human thus belongs to the reflector class. Since reflectors have little variation, relatively few samples of this class was collected. Instead the data size was synthetically increased by duplicating samples and adding different noise components to simulate different SNRs. The six drones were a DJI matrice 200 v2 (D1), a DJI Mavic 2 Pro (D2), a DJI Phantom 3 (D3), a custom built FPV drone (D4), a custom built hexacopter (D5), and a Syma X23W toy drone (D6). Details of the different classes are shown in Table 6.1. The distribution of the estimated RCS for the different targets are shown in Fig. 6.2. We see that discriminating drones from birds based on RCS alone will have poor classification accuracy.

The data was randomly divided into a test set of 1000 samples, a validation set of 1000 samples and the remaining were used as training samples, 4028 to 10093 depending on class. See Table 6.1. The variation in range to the targets, 5 to 200 meters depending on target and scenario, together with different RCS distributions (see Fig. 6.2), resulted in uneven distributions in the SNR. Unless compensated for, this will cause the classifier to bias the estimate according to the SNR distribution. As this is contrary to our objective of performing the classification based on the shape of the spectrum alone, we first perform a normalization of the SNR distributions.

Table 6.1: Datasets, available at [143].

Class	Description	Samples
D1	Type: DJI Matrice 200 v2 Diagonal Wheelbase: 643 mm Propeller Size: $17''$ (430 mm)	6921
D2	Type: DJI Mavic 2 Pro Diagonal Wheelbase: 354 mm Propeller Size: $8.7''$ (220 mm)	9555
D3	Type: DJI Phantom 3 Diagonal Wheelbase: 350 mm Propeller Size: $9''$ (227 mm)	10212
D4	Type: Custom built FPV drone Diagonal Wheelbase: 300 mm Propeller Size: $7''$ (178 mm)	8735
D5	Type: Custom built with a Tarot 680 Pro frame Diagonal Wheelbase: 680 mm Propeller Size: $12''$ (305 mm)	12093
D6	Type: Syma X23W Diagonal Wheelbase: 155 mm Propeller Size: $3.8''$ (97 mm)	11252
H	Different humans walking and running	6028
B	Flying birds, with and without wing flaps. The data consists of black-headed gulls (58.8%), seagulls (35.6%), herons (5%), pigeons (0.4%), and raven (0.2%).	7792
R	Trihedral corner reflectors	3280



The data is normalized to have a uniform SNR distribution between 3 and 30 dB after range compression. This is done by adding complex Gaussian noise to samples that have a higher SNR than a desired target SNR. The target SNR is uniformly sampled in dB scale. A sample is in some cases used several times but with different AWGN components to generate 10000 training samples, 4000 validation samples and 4000 test samples in each class. Samples are only reused within their respective set. To prevent classification bias on the different

distributions in velocity, all samples are also normalized to have zero bulk-Doppler.

Each data sample $\boldsymbol{S} = [\boldsymbol{s}_1, ..., \boldsymbol{s}_5]^T \in \mathbb{C}^{5\times 150}$ is a segment of a scan. These consist of five samples in range, corresponding to about 5 meters, and 150 samples in azimuth, corresponding to about 2.3° or 9 ms in time, of which 4 ms is within the beamwidth of 1°. The micro-Doppler spectrum at each range is obtained as

$$\boldsymbol{x}_r = |\mathbf{F}\,(\boldsymbol{\beta}_3 \circ \boldsymbol{s}_r)\,|^2 \tag{6.1}$$

for $r = 1, 2, ..., 5$, where $\mathbf{F} \in \mathbb{C}^{150\times 150}$ is the DFT matrix and $\boldsymbol{\beta}_3 \in \mathbb{C}^{150\times 1}$ is a complex steering vector that puts the spectrum peak, or bulk-Doppler, of $\boldsymbol{x}_3$ at frequency zero. We then normalize and scale $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_5]^T$ as

$$\boldsymbol{X}' = \log\left(\boldsymbol{X}/\bar{x}\right) \tag{6.2}$$

where $\bar{x}$ is the mean of all elements in $\boldsymbol{X}$. The stated SNRs always refer to the SNR at the center, or peak, of $\boldsymbol{s}_3$, i.e. before conversion to frequency domain.

6.2.2 Synthetic Data

The model for generating synthetic data is based on the propeller model in [144] and [145] with some modifications. Similar methods for modeling micro-Doppler spectra are presented in [146–150]. The simulated return signal is given as

$$\tilde{s}_{r,p} = G_{r,p}\,(1 + \rho z_p/\bar{z}) + w_{r,p} \tag{6.3}$$

where

- z_p is the combined propeller return at pulse/FM-sweep p
- $w_{r,p}$ is zero mean complex valued AWGN
- $G_{r,p}$ is the combined pulse gain at range index r and azimuth beam gain at index p. This is tailored to match the used sensor
- $\bar{z}$ is the RMS value of $\boldsymbol{z}$ over all p pulses
- ρ sets the strength of the propellers relative to the drone body, which has unit strength.

The indices r and p are defined such that the center indices is at the peak of the pulse and beam shape. The SNR is determined by the variance in $w_{r,p}$. The combined

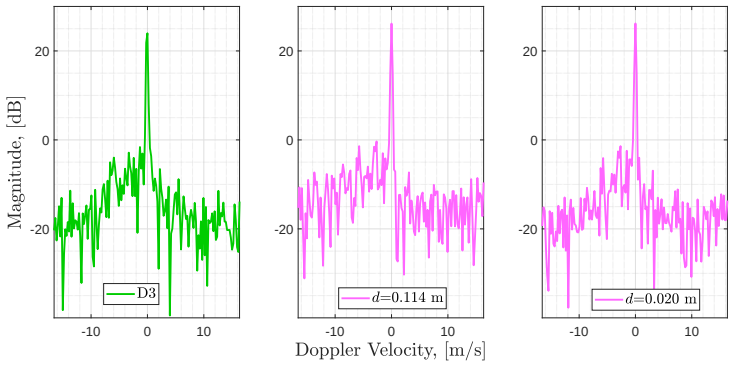


Figure 6.3: Example of parameter selection in the drone model. The three plots show a real measurement from D3, a synthetic sample with $d = 114$ mm, i.e. the true blade length for D3, and a synthetic sample with $d = 20$ mm. The shown view corresponds to $\boldsymbol{x}_3$ in (6.1). By reducing the blade length, and consequently the model bandwidth, we obtain a synthetic sample that appears to have a spectrum pattern that is more consistent with the real measurement of D3. The other parameter values for the two synthetic drone examples were $\nu = 100/17000$, $\rho = -15$ dB, $N_p = 4$, $\alpha_1 = 0.7$, and SNR$= 25$dB.

propeller return is

$$z_p = \sum_{n=1}^{N_p} \sum_{i=1}^{N_s} \sum_{k=1}^{2} \alpha_k \exp\left(j(-1)^k I(\theta_{n,i})\theta_{n,i}\right) \operatorname{sinc}(\theta_{n,i}) \tag{6.4}$$

$$\theta_{n,i} = 2\pi d_i \sin\left(2\pi\nu p + \phi_n\right)/\lambda \tag{6.5}$$

$$N_s = \lceil 10d/\lambda \rceil \tag{6.6}$$

$$d_i = di/N_s \tag{6.7}$$

$$I(\theta) = \begin{cases} 1, & \text{if } \cos(\theta) \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{6.8}$$

where N_p is the number of propellers, λ is the carrier wavelength, ν is the propeller's rotation rate in rounds per second (RPS) divided by the PRF, ϕ_n is the initial phase of propeller n, and $\operatorname{sinc}(\cdot)$ is the non-normalized sinc function.

The model comes from regarding a propeller blade return as the sum of the return from N_s cylinders with linearly increasing length from d/N_s to d, a simple approach to account for a propellers tapered form. N_s was chosen such that each such cylinder increased by a fraction of a wavelength. The parameters α_1

and $\alpha_2 = 1 - \alpha_1$ allow us to scale the returns from the two blades on each propeller differently. The propeller blade that moves towards the radar, and the blade that moves away from the radar may have different RCS values from the radar perspective, resulting in an asymmetric phase change. This can be seen in Fig. 6.8, where the drone micro-Doppler is often skewed towards negative frequencies, corresponding to $\alpha_1 > \alpha_2$ on average.

The blade length d in the model had to be considerably shorter than actual blade lengths in order to obtain a better match between synthetic and real data. An example of this is shown in Fig. 6.3. As discussed in Section 2.7, our intention with this model is not to perfectly explain real data, but rather to shape ANNs classifiers in a way that makes them generalise better to unknown drone classes. Deviations from physical parameter values such as blade lengths of a few centimeters is therefore not a problem in this case. One could also argue that it is the root of the blade that has the strongest reflections, and the choice of a short propeller length would thus be consistent with this assumption.

A synthetic data sample $\tilde{\boldsymbol{S}} = [\tilde{\boldsymbol{s}}_1, ..., \tilde{\boldsymbol{s}}_5]$, where $\tilde{\boldsymbol{s}}_r = [\tilde{s}_{r,1}, ..., \tilde{s}_{r,150}]^T$, is normalized in the same manner as a real sample using (6.1) and (6.2). Parameter values used when generating synthetic data are shown in Table 6.2 and Table 6.4. Examples of micro-Doppler spectra of real and simulated targets are shown in Fig. 6.4.

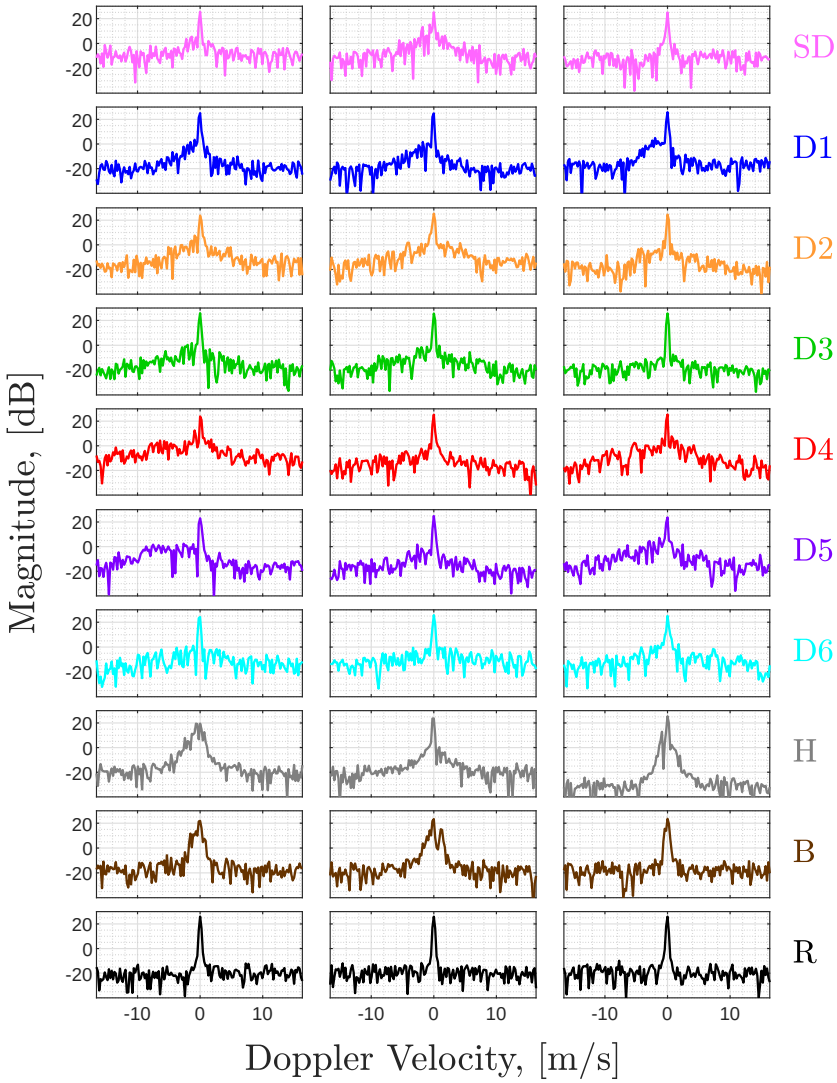


Figure 6.4: Examples of the micro-Doppler spectra for the nine classes in Table 6.1, including three synthetic data samples from the presented model labeled SD, top row. The shown view corresponds to $\boldsymbol{x}_3$ in (6.1). All samples have approximately 25 dB SNR.

Table 6.2: Model Parameters for Synthetic Data

Parameter	Uniform Distribution Limits
ϕ	[0, 2π]
ν	[10, 200]/17000
$d\,[m]$	[0.001, 0.02]
N_p	[1, 10]
ρ [dB]	[-20, 0]
α_1	[0.25, 1]
SNR [dB]	[3, 30]

6.3 Approach One: Model-Aided Classification

6.3.1 Classifier Model

When evaluating the classifier on out-of-distribution drones, one out of the six drone classes is used for training together with the three non-drone classes. The data from the five other drone classes are added to the test set. To improve the classifier's robustness to out-of-distribution drones, we also train a classifier with an additional class consisting of simulated data using the model in Section 6.2.2. The parameter values for this are shown in Table 6.2.

The value of the number of classes N_c thus depends on the scenario. When evaluating out-of-distribution drones $N_c = 4$, i.e. one drone class and the three non-drone classes, or $N_c = 5$ when including the synthetic drone class. Since we have six drone classes we have six different scenarios of "known" drones. To benchmark the problem we also train on all six drone classes. We do this without synthetic data, i.e. $N_c = 9$, and with synthetic data, $N_c = 10$. The identification step is done by assigning target type as

$$\hat{C} = \underset{C}{\text{argmax}} \;\; \hat{\boldsymbol{y}}[C] \tag{6.9}$$

where $\hat{\boldsymbol{y}}$ is the CNN output. If $\hat{C}$ represents one of the drone classes the target is classified as drone, otherwise as non-drone.

The classifier consists of a five layer CNN. The architecture is shown in Table 6.3. It was found that more layers did not improve the performance or even reduced the performance. This type of sensitivity to noisy data with deep networks is also reported in [142]. To reduce bias towards one of the classes (drone/non-drone), we balance the datasets such that the total number of non-drone samples is always equal to the total number of drone samples. We do this by duplicating samples in the class with fewer samples. The classifier was trained

Table 6.3: Neural Network Architecture

Layer Type	Weight Shape	Strides	Output Shape	Activation
Convolutional	F: 10, K: 5×8	[1,1]	$1 \times 143 \times 10$	ReLu
Convolutional	F: 10, K: 1×8	[1,2]	$1 \times 68 \times 10$	ReLu
Convolutional	F: 10, K: 1×8	[1,3]	$1 \times 21 \times 10$	ReLu
Fully Connected	32×210	-	32×1	ReLu
Fully Connected	$N_c \times 32$	-	$N_c \times 1$	Softmax

F: Number of filters/kernels. K: Kernel size. Total number of parameters: $8782 + N_c(32 + 1)$.

using stochastic gradient descent with a batch size of 128 and Adam optimisation. The loss function was the categorical cross-entropy loss with one-hot encoded labels, see Section 3.3.1. Due to the non-convex optimization and random initial values of the weights, slightly different solutions are obtained each training run. We therefore train ten networks and choose the one with the best classification accuracy on the validation set. Each network was trained for 25 epochs, allowing the validation accuracy to converge.

6.3.2 Results

All the presented results concern the test data. The confusion matrix when trained on all nine classes is shown in Fig. 6.5. The mean identification accuracy over all classes was 0.68. In the top plot in Fig. 6.7 we see the classification accuracy per class when trained on only one drone as well as all drone classes. In the latter case the mean classification accuracy was 0.88 for drones and 0.93 for non-drones. When trained on only one class the accuracy was high for the known drone class, > 0.9, but varied more for the other out-of-distribution drones with a mean accuracy of 0.78 and down to as low as 0.35, i.e. worse than guessing. In the bottom plot in Fig. 6.7 we see the effect of introducing synthetic data. Many drone samples that would otherwise have been classified to one of the non-drone classes are now classified to the synthetic drone class. This yields an improvement in out-of-distribution classification accuracies by as much as 123% (0.35 to 0.78), with a mean improvement of 10% (0.78 to 0.86).

The presented accuracies are the mean over the entire SNR range, 3-30 dB. The mean classification accuracy as a function of SNR is shown in Fig. 6.6 where we see that for SNR> 17.5 dB the mean accuarcy is 95% or more when all drones are used in training. Looking at the same SNR limit and cases when only one drone class is used, the mean accuracy on the known classes is also $> 95\%$ whether synthetic data is used or not. The accuracy on the out-of-distribution drones however is

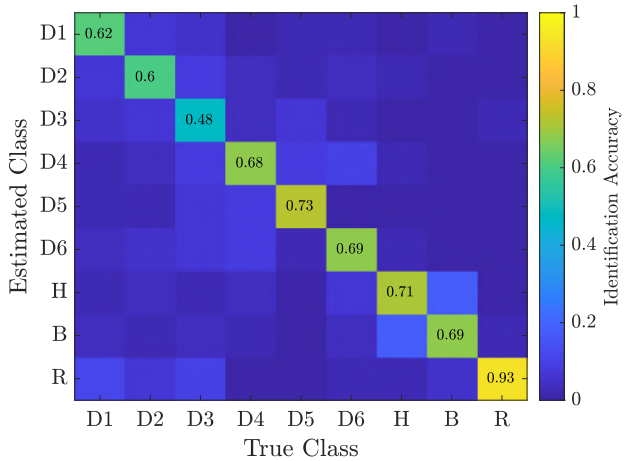


Figure 6.5: Confusion matrix for the nine classes with mean identification accuracy 0.68. The classification accuracy for each class can be seen in the bottom rows in Fig. 6.7. The total classification accuracy as a function of SNR can be seen in Fig. 6.6 as the black solid curve.

only $> 82\%$ without the synthetic data class and $> 91\%$ with synthetic data. For reference we also show the results when using only simulated drone data.

As described in Section 3.3.8, we examine the data dependent equivalent filters of the CNN. Ignoring the final softmax activation, the output of the CNN can be written as an affine data dependent transformation for a given input, i.e.

$$\hat{\boldsymbol{y}}_n = \tilde{\boldsymbol{W}}_n \text{vec}\left(\boldsymbol{X}'_n\right) + \tilde{\boldsymbol{b}}_n \tag{6.10}$$

where $\tilde{\boldsymbol{W}}_n \in \mathbb{R}^{N_c \times 750}$ and depends on $\boldsymbol{X}'_n$, see (3.43)-(3.47). We can regard a row C in $\tilde{\boldsymbol{W}}_n$ together with element C in $\tilde{\boldsymbol{b}}$ as an affine filter for class C that is designed through the optimization, to yield a high scalar output if $\boldsymbol{X}'_n$ belongs to class C and relatively low otherwise.

As with our experiment on digit classification in Section 3.3.8, we can gain some insight into what the classifier bases its decision on by looking at these filters. To simplify this we focus on the part of $\tilde{\boldsymbol{W}}_n$ that corresponds to the third (i.e. center) range row in the input. We also only look at the mean over each class. We denote these as $\bar{w}_3^+$ when the input belongs to the same class as $\bar{w}_3$ and $\bar{w}_3^-$ when it does not. We thus have for each class one $\bar{w}_3^+$, and eight cases of $\bar{w}_3^-$. These are shown in Fig. 6.8 together with the mean of the third range row of the input, $\bar{\boldsymbol{x}}'_3$. We see that for the non-drone classes H,B and R, the CNN puts higher weight to low frequency

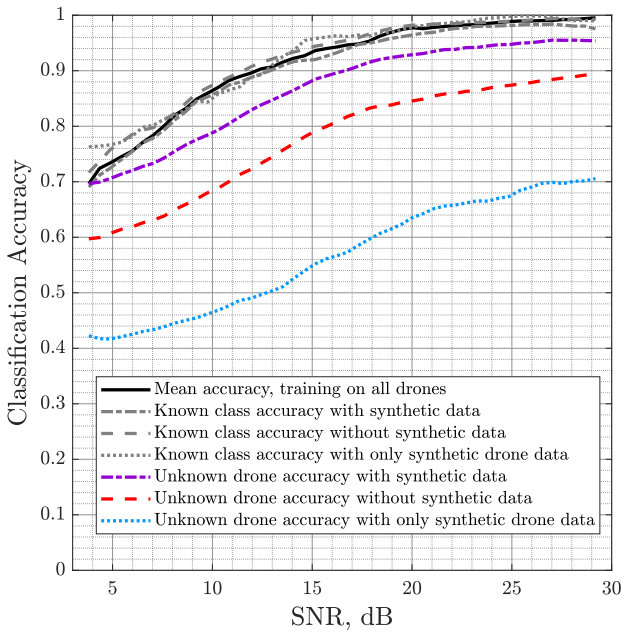


Figure 6.6: Mean classification accuracy as a function of SNR. Samples that are represented in the training data all follow the same trend. Out-of-distribution, or unknown, drone samples have lower accuracies, although a clear improvement can be seen when synthetic data is used in the training process. For reference we also show the case when only synthetic drone data is used in training. In this case the known classes are all non-drones and the unknown classes are the six drones.

components compared to the six drone classes, at least when the input belongs to the same class as the adaptive filter (w_3^+).

We also note that the filters of several drone classes are more asymmetric in frequency compared to non-drone classes. This is a good sanity check since one can expect a larger and skewed bandwidth to be an important discriminating factor, e.g. some of the handcrafted features presented in [151] and used in [131] are of a similar nature. We also see that the filter for drone D2 has high weights at ± 5 m/s which appear to correspond to peaks at the same velocities in $\bar{\boldsymbol{x}}_3'$.

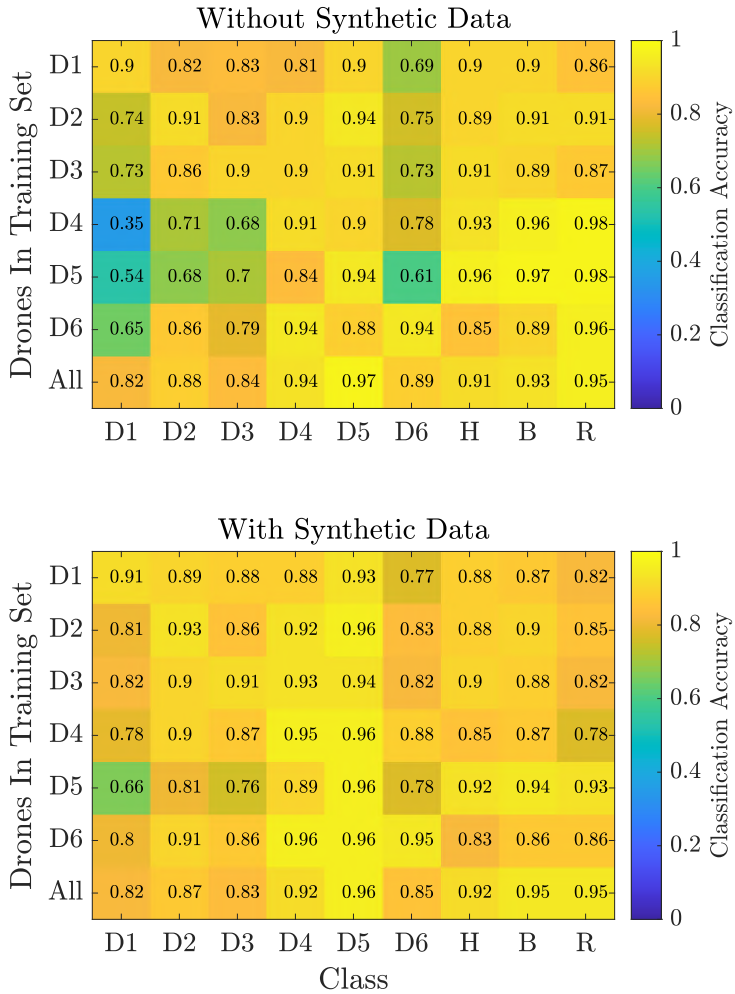


Figure 6.7: Classification accuracy for the nine classes for a classifier trained on only one of the six classes, row 1-6, and all drones, bottom row. Top: no synthetic data. Bottom: with the synthetic data class. The accuracies on drone 1 and drone 6 are relatively poor when these belong to the out-of-distribution drones and a notable increase in accuracy can be seen for these drones when the synthetic data class is used.

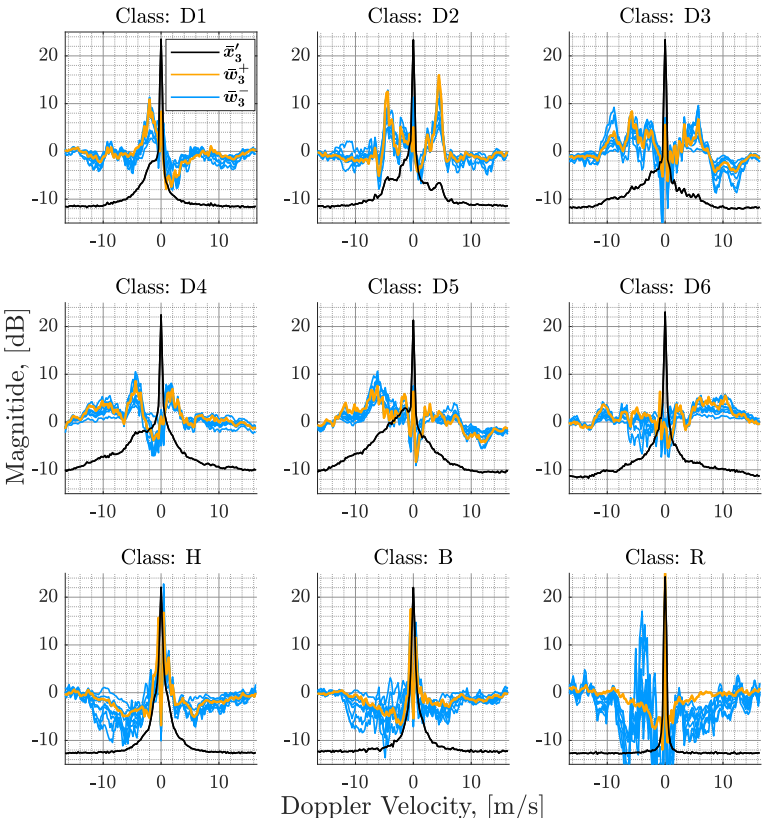


Figure 6.8: Mean of all equivalent filters. For each of the nine classes the blue and orange curves are the class-means of each filter in $\boldsymbol{W}_n$ for the third range row. In the legend the superfix "$+$" denotes that the input class matches the filters class and "$-$" denotes the opposite. The mean of the third row in the input $\boldsymbol{X}'$, labeled $\boldsymbol{x}'_3$, is also shown for each class. For visualisation purposes the data is scaled by $10\log_{10}(e)$ to convert the y-axis to dB scale. All filter weights are also scaled by an additional factor of 10 for easier comparison with the mean of $\boldsymbol{x}'_3$.

Table 6.4: Model Parameters for Synthetic Data

Parameter	Uniform Distribution Limits
ϕ	[0, 2π]
ν	[10, 200]/17000
δ	[0.5, 1.5]
$d\,[m]$	[0.001, 0.02]
N_p	[1, 10]
ρ [dB]	[-20, 0]
α_1	[0, 1]
SNR [dB]	[20, 30]

6.4 Approach Two: Anomaly Based Classification

6.4.1 Classifier Model

The classifier and anomaly detector is constructed in two steps. First we train the CNN based autoencoder on data generated from the model in Section 6.2.2, but with a slight modification. The normalized propeller RPS ν in (6.5) is scaled as $\delta_n \nu$ where δ_n varies for the n propellers while ν is the same. We thus "jitter" the RPS value of each propeller. This is more consistent with RPS values of real drone propellers. The parameter values used for data generation are shown in Table 6.4.

The length of the encoding in the autoencoder, or dimension k of the latent space Z, should be small enough such that the Euclidean distance between two points in the latent space is meaningful, and at the same time large enough to sufficiently encode the data. We set $k = 4$ based on empirical testing, see Section 6.4.2. Our encoder and decoder functions are thus

$$\boldsymbol{z} = f_E(\boldsymbol{X}') \tag{6.11}$$

$$\hat{\boldsymbol{x}}_3 = f_D(\boldsymbol{z}) \tag{6.12}$$

which we have trained using the MSE loss between $\boldsymbol{x}_3$ and $\hat{\boldsymbol{x}}_3$, i.e. the range bin where the signal is the strongest. We train the autoencoder on 80,000 samples and use 20,000 samples for validation. The architecture of the autoencoder network is shown in Table 6.5.

In the second step we evaluate the encoder on N_{tr} training samples on each of the three non-drone, or normal, classes: H, B and R. For this we skip the final ReLu activation. We also add complex Gaussian white noise of sufficient power to samples in the time domain, i.e. $\boldsymbol{S}$, to achieve different SNRs, γ. The upper limit of our SNR range is thus determined by the higher SNRs of our real radar

Table 6.5: Neural Network Architecture

Encoder Network, f_E				
Layer Type	Weight Shape	Strides	Output Shape	Activation
Convolutional	F: 64, K: 5×5	[1,2]	$1 \times 73 \times 64$	ReLu
Convolutional	F: 32, K: 1×5	[1,2]	$1 \times 35 \times 32$	ReLu
Convolutional	F: 16, K: 1×5	[1,2]	$1 \times 16 \times 16$	ReLu
Fully Connected	4×256	-	4×1	ReLu
Decoder Network, f_D				
Layer Type	Weight Shape	Strides	Output Shape	Activation
Fully Connected	32×4	-	32×1	ReLu
Fully Connected	64×32	-	64×1	ReLu
Fully Connected	150×64	-	150×1	Linear

F: Number of filters/kernels. K: Kernel size. Total number of parameters in Encoder: 15797, Decoder: 12054

samples. This gives us $\boldsymbol{z}^{\mathrm{H},\gamma}, \boldsymbol{z}^{\mathrm{B},\gamma}, \boldsymbol{z}^{\mathrm{R},\gamma} \in \mathbb{R}^{4\times N_{tr}}$. For each class and SNR, we form one Gaussian model using the maximum likelihood estimate of the mean and covariance matrix,

$$\boldsymbol{\mu}^{\mathrm{c},\gamma} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \boldsymbol{z}_i^{\mathrm{c},\gamma} \tag{6.13}$$

$$\boldsymbol{\Sigma}^{\mathrm{c},\gamma} = \frac{1}{N_{tr}-1} \sum_{i=1}^{N_{tr}} (\boldsymbol{z}_i^{\mathrm{c},\gamma} - \boldsymbol{\mu}^{\mathrm{c},\gamma})(\boldsymbol{z}_i^{\mathrm{c},\gamma} - \boldsymbol{\mu}^{\mathrm{c},\gamma})^T \tag{6.14}$$

for c=H,B,R. We set $N_{tr} = 50$ samples for each class. For the classification step, we define a log likelihood score function as

$$\begin{aligned} t(\bar{\boldsymbol{z}}, \boldsymbol{\mu}^{\mathrm{c},\gamma}, \boldsymbol{\Sigma}^{\mathrm{c},\gamma}) = &- (\bar{\boldsymbol{z}} - \boldsymbol{\mu}^{\mathrm{c},\gamma})^T (\boldsymbol{\Sigma}^{\mathrm{c},\gamma})^{-1} (\bar{\boldsymbol{z}} - \boldsymbol{\mu}^{\mathrm{c},\gamma}) \\ &- \log(|\boldsymbol{\Sigma}^{\mathrm{c},\gamma}|) + \beta\eta^{c,\gamma}(\bar{\boldsymbol{z}}). \end{aligned} \tag{6.15}$$

The last term, $\beta\eta^{c,\gamma}(\bar{\boldsymbol{z}})$, acts as a prior and is motivated as follows. We may assume that drones will have a relatively broad micro-Doppler spectrum. We can generate samples of this by fixing ν, d and ρ to the highest value in their respective range in Table 6.4. We call this specific synthetic data class D0. We then project these samples unto Z and take the mean, giving us the point $\boldsymbol{\mu}^{\mathrm{D0}} \in \mathbb{R}^{4\times 1}$. A test sample that is close to this point can then be considered more likely to be a drone than a

sample that is far away from this point. Let

$$\boldsymbol{v}_{\mathrm{D0}-c} = \boldsymbol{\mu}^{\mathrm{D0}} - \boldsymbol{\mu}^{c,\gamma} \tag{6.16}$$

$$\boldsymbol{v}_{\bar{z}-c} = \bar{\boldsymbol{z}} - \boldsymbol{\mu}^{c,\gamma} \tag{6.17}$$

be the vectors from the mean of a known class c to $\boldsymbol{\mu}^{\mathrm{D0}}$, (6.16), and the vector from the mean of the known class to a sample $\bar{\boldsymbol{z}}$, (6.17). The normalized projection of $\boldsymbol{v}_{\bar{z}-c}$ onto $\boldsymbol{v}_{\mathrm{D0}-c}$ will be a value between -1 and 1, which we scale as

$$\eta^{c,\gamma}(\bar{\boldsymbol{z}}) = \frac{1}{2}\left(1 - \frac{\boldsymbol{v}_{\mathrm{D0}-c} \cdot \boldsymbol{v}_{\bar{z}-c}}{||\boldsymbol{v}_{\mathrm{D0}-c}||_2 ||\boldsymbol{v}_{\bar{z}-c}||_2}\right) \tag{6.18}$$

such that $\eta^{c,\gamma}(\bar{\boldsymbol{z}}) \in [0, 1]$. By adding $\eta^{c,\gamma}(\bar{\boldsymbol{z}})$ to the score in (6.15) we increase the score if $\boldsymbol{v}_{\bar{z}-c}$ points away from $\boldsymbol{\mu}^{D0}$ and give zero increase if the sample $\bar{\boldsymbol{z}}$ lies on a line between $\boldsymbol{\mu}^{c,\gamma}$ and $\boldsymbol{\mu}^{\mathrm{D0}}$. To adjust the effect of this added score, we scale it by the parameter β.We select a classifier depending on the sample SNR and classify a candidate sample $\bar{\boldsymbol{z}}$ as

$$\hat{\mathrm{c}} = \begin{cases} \hat{\mathrm{c}}', & \text{if } t\left(\bar{\boldsymbol{z}}, \boldsymbol{\mu}^{\hat{\mathrm{c}}',\gamma}, \boldsymbol{\Sigma}^{\hat{\mathrm{c}}',\gamma}\right) \geq T_M \\ \text{anomaly}, & \text{otherwise} \end{cases} \tag{6.19}$$

where

$$\hat{\mathrm{c}}' = \underset{\mathrm{c}}{\operatorname{argmax}}\ t(\bar{\boldsymbol{z}}, \boldsymbol{\mu}^{\mathrm{c},\gamma}, \boldsymbol{\Sigma}^{\mathrm{c},\gamma}) \tag{6.20}$$

and $\bar{\boldsymbol{z}}$ is the mean of M consecutive samples. T_M is a threshold that depends on M.

Apart from the aforementioned benefit of not relying on any specific drone for training the classifier, we also note some other advantages of this setup. First, since we train the encoder on simulated data, we have no limitations on the amount of data. Second, to estimate the low dimensional mean vectors and covariance matrices required for the modified quadratic discriminant analysis (QDA) classifier, we only need a relatively small sample size of real samples. Third, we can easily add new target types without having to re-train the entire classifier. This is very difficult to do using fully neural network based classifiers. If, e.g., a target is consistently classified as an anomaly we can take the obtained data and form a new Gaussian model for that target. Fourth, we note that the Gaussian model is only an approximation of the distribution of the encoded samples. However, from the central limit theorem we have that an average of sufficiently many samples will eventually converge to a Gaussian distribution, giving stronger theoretical support to this approach.

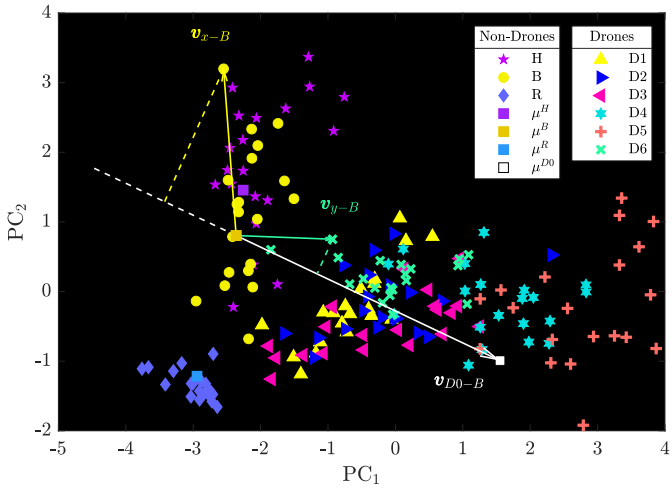


Figure 6.9: Projection of the latent space Z with 20 points for each of the nine classes. This has been projected onto two dimensions using PCA for illustrative purposes. The non-drone samples and drone samples appear in different regions in the image. The mean points of the three non-drone normal classes, as well as the point of the hypothesis drone $\boldsymbol{\mu}^{\text{D0}}$ are also shown. We have listed $\boldsymbol{\mu}^{\text{D0}}$ as a non-drone sample since it is obtained from synthetic data and therefore not an actual drone. The projection of $\boldsymbol{v}_{y-B}$ onto $\boldsymbol{v}_{\text{D0}-\text{B}}$ will have a positive value yielding a lower score in (6.15) via (6.18). The projection of $\boldsymbol{v}_{x-B}$ onto $\boldsymbol{v}_{\text{D0}-\text{B}}$ will on the other hand be negative, yielding a higher score. The sample at $\boldsymbol{v}_{x-B}$ may thus be classified to B (or possibly H) while the sample at $\boldsymbol{v}_{y-B}$ may be classified as an anomaly, despite $\boldsymbol{v}_{y-B}$ being closer to $\boldsymbol{\mu}^{\text{B}}$ than $\boldsymbol{v}_{x-B}$.

6.4.2 Results

An illustration of the latent space obtained with the CNN is shown in Fig. 6.9, where the four dimensions of $\bar{\boldsymbol{z}}$ have been projected onto two dimensions using PCA. The non-drone samples and drone samples appear in different regions in the image indicating that the CNN autoencoder is suitable for classification. We also see that different drones appear in different regions to some extent, indicating the possibility of drone identification, i.e discriminating between different drones. This is however beyond the scope of the anomaly based approach considered here. Fig. 6.9 also illustrates the concept of the introduced prior in the score function in (6.15). The point $\boldsymbol{\mu}^{\text{D0}}$ was obtained from 1000 synthetic samples.

The dimension of the latent space was chosen to $k = 4$. There are some

Table 6.6: Latent Space Dimensionality
Results from 20 runs

k	**K=3**	**K=4**	**K=5**	**K=6**	**K=7**	**Mean Loss: ReLu**	**Mean Loss: Sigmoid**
0	3	1	0	0	0	3.69	-
1	1	0	3	0	0	2.18	-
2	7	3	1	0	0	1.91	-
3	9	13	6	6	2	1.75	1.94
4	-	3	5	8	5	1.68	1.84
5	-	-	5	5	10	1.66	1.79
6	-	-	-	1	3	1.64	1.78
7	-	-	-	-	0	-	1.75

interesting observations to be made regarding this choice. The ReLu activations used in the CNN have a tendency to induce sparsity, or in other words some nodes will always be zero valued and can thus be removed [152]. This is of particular interest in autoencoders since we aim to find compact representations of the data. Table 6.6 shows the number of active nodes after training, or the dimension k, for different cases where the bottleneck layer in the CNN was set to K nodes. The results are from 20 different instances of training. We see that in almost all cases the number of active nodes k is less than K.

This induced sparsity comes as a consequence of using ReLu activations and is obtained without any additional regularisation, such as using the L1-norm [153]. Running the same experiment as in Table 6.6 with sigmoid activations gave $k = K$ in all 20 runs for all values of K. Since the sigmoid function is never exactly zero, we considered a node as inactive if it was less than some value ε for all validation samples. Even with a rather liberal limit of $\varepsilon = 0.5$ we still got $k = K$ in all runs. Apart from induced sparsity, we also note in Table 6.6 that the ReLu activation yields lower loss than the sigmoid activation for a given k.

We also note that the mean loss decreases with k but the rate of decrease is limited beyond $k = 4$ when using ReLu activations. In the network used in this Section, we set $K = 5$ and obtained $k = 4$ active nodes in the latent space. The MSE loss for the autoencoder was, after 25 epochs, 1.68 for both training and validation data. For reference, the same loss when using PCA on four dimensions was 2.44 and the loss using the mean, i.e. the mean variance, was 3.77.

When estimating the Gaussian models we got slightly different results depending on which N_{tr} samples we selected. The results presented in this section are therefor an average of 100 trials.

In Fig. 6.10 we show the mean accuracy for drone and non-drone targets as

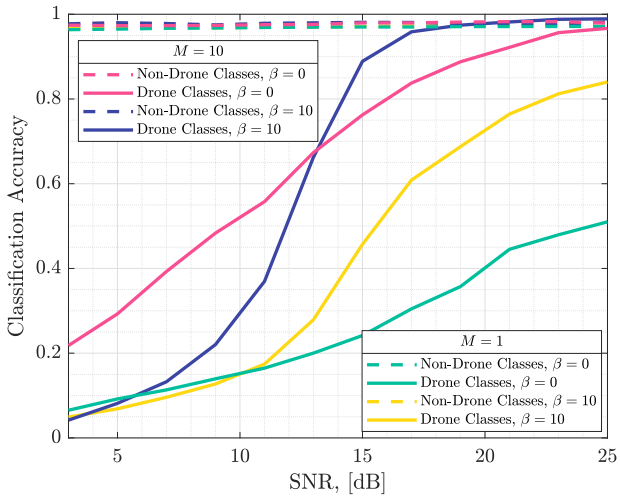


Figure 6.10: Mean classification accuracy for drone and non-drone classes as a function of SNR. For all cases the threshold T_M was set such that the classification accuracy on the three non-drone classes was at least 95%. We see that the accuracy increases dramatically when samples are integrated, i.e. when M=10. At high SNRs we obtain an additional increase in accuracy when setting $\beta = 10$.

a function of SNR. Here we have set the threshold T_M such that the non-drone samples are classified as such with at least 95% empirical probability. Fig. 6.10 also illustrates the effect of adding the prior in (6.18) to the score in (6.15). Setting $\beta = 0$, equivalent to not having any prior, gives poorer accuracy at high SNRs, and slightly better accuracy at low SNRs when compared to $\beta = 10$. The optimal value of β thus depends on which SNR one wants to optimize against. Increasing β to 100 gives approximately the same results but with a greater reduction in accuracy at lower SNR values and a slight increase at high SNRs for the drone classes.

In Fig. 6.11 we show the ROC curve for the case when the SNR=20 dB and $\beta = 10$. In this case the number of test samples was 4430, 9160, 6830, 7370, 8760 and 7070 for D1-D6 respectively, and 5210, 1450 and 3150 for H,B and R respectively. Note that we add noise to real data to produce the desired SNR. This limits the number of test samples we can use as we increase the SNR. We again note in Fig. 6.11 a significant improvement in the area under receiver operating characteristic curve (AUC) metric when we integrate ten samples.

To illustrate the gain of using CNNs in the classifier, we compare this to a simpler but perhaps more intuitive method based on the bandwidth of the data

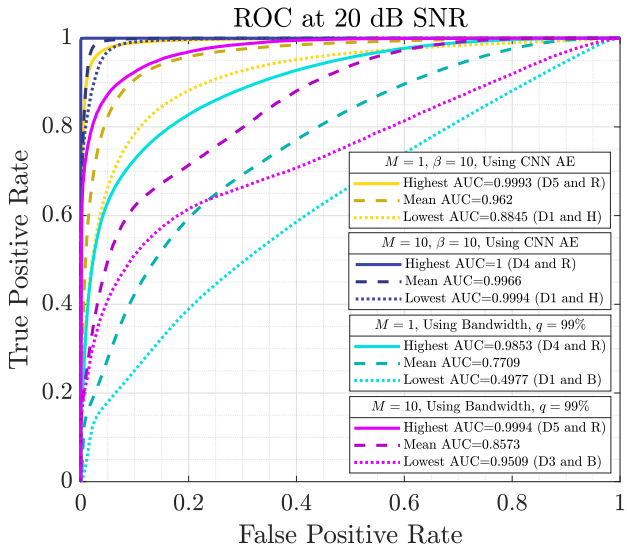


Figure 6.11: ROC curves where a true positive is a drone classified as an anomaly, and a false positive is a non-drone classified as an anomaly. We show the best, worst and mean curve in regards to the AUC. In all cases we set $\beta = 10$. In the legend, AE stands for autoencoder.

samples. We estimate the q-percentile bandwidth of the positive and negative frequencies, meaning the bandwidth that contains $q\%$ of the energy over positive frequencies and $q\%$ of the energy over negative frequencies. This gives a two element vector for each sample. Since the spectrum is not always symmetric we estimate the positive and negative frequencies separately. We can then construct a classifier based on this two element vector in the same way as we did on $\bar{\boldsymbol{z}}$. In this case $\boldsymbol{\mu}^{\mathrm{D0}}$ was estimated from the bandwidth of synthetic samples. The ROC results for this is shown in Fig. 6.11. Here $q = 99\%$ was chosen since this gave the highest mean AUC. We note that for certain cases this method works quite well but not in general, whereas the CNN based method has a smaller spread between the highest and lowest AUC values. The comparison in Fig. 6.11 is not entirely fair since we only use two dimensions in the bandwidth approach and the CNN autoencoder has four. The CNN autoencoder approach is however still superior even if we reduce k to two.

Next we compare the CNN to linear PCA with and without the use of model data. Table 6.7 shows the accuracy for each class when we select the threshold that yields the highest mean accuracy in the ROC when the SNR = 20 dB and $M = 10$,

Table 6.7: Classification Accuracy, SNR=20 dB, M=10

Class	D1	D2	D3	D4	D5	D6	H	B	R
CNN[1]	**0.93**	**0.96**	**0.95**	**1.0**	**1.0**	**0.98**	**0.99**	**1.0**	**1.0**
CNN[2]	0.19	0.50	0.35	0.89	0.96	0.40	0.92	1.0	0.97
PCA[1]	0.41	0.66	0.60	0.97	0.99	0.96	0.87	0.99	0.98
PCA[2]	0.41	0.71	0.45	0.93	0.95	0.81	0.40	0.82	1.0

[1] obtained from synthetic data. [2] obtained from real data from the three normal classes H,B and R.

for four different cases. The first, CNN[1] is the presented CNN solution. In the second case, which we refer to as CNN[2], the CNN autoencoder is only trained on the $3 \times N_{tr}$ non-drone training samples and no model is used. This also excludes the prior in (6.18) since this uses the model. The third, PCA[1], uses an autoencoder obtained from PCA on model data with $\beta = 10$. The fourth, PCA[2], only uses PCA on the $3 \times N_{tr}$ non-drone training samples. We note a benefit in terms of accuracy when using the nonlinear CNN encoder trained on model data. The average accuracy is in this case 97% for the drones and 99% for the non-drones.

6.5 Conclusions

The presented methods are motivated by two main factors. The first is the problem of not having examples of all possible drones, which will arguably always be the case in practice. We have highlighted the merits of using a model in such cases. Secondly, we consider the problem of having a limited amount of labelled data in general. We have on a limited scale evaluated a small CNN classifier's ability to accurately classify "out-of-distribution" drones. We have done this with two different approaches.

In the first approach, we saw that the accuracy on the unknown drones varied between worse than guessing to similar accuracies as when all drones are represented in the training data. By introducing a synthetic drone class with data generated from a mathematical drone model, the accuracies of the unknown drones reached a more uniform level, still with a slight reduction compared to the ideal case where all classes are known. We also saw that the CNN appears to weight different parts of the spectrum depending on class, a behavior that goes in line with known theory. When training on all data classes the classification accuracy approaches 100% for high SNRs, a high accuracy, especially considering only 9 ms dwell time is used for classification, of which 4 ms is within the 3 dB beamwidth.

In the second approach we have gone to the extreme of assuming that no real drone examples are available. This approach is, as far as we know, not taken

by any other study on drone classification using micro-Doppler radar and neural networks. We have thus addressed the question "What if another type of drone, not represented in the training data, is observed?" since no drone is represented in the training. By utilising a model when constructing the classifier, we only need real data samples when estimating mean vectors and covariance matrices in a low dimensional space. This significantly reduces the required number of real data samples for each known class. In the presented results we have used 50 samples for each of the tree normal classes, H, B and R. We have showed that whether we use a CNN or PCA based autoencoder, we obtain poorer accuracy when using real data alone than what is obtained when instead training on simulated data from the presented model.

The second approach yields poorer classification accuracies, especially at lower SNR values. We see this when comparing Fig. 6.6 and Fig 6.10. By temporally averaging the encoder projections, the accuracy of the classifier was improved. Averaging over 10 samples, corresponding to an effective observation time of 90 ms, gave at 20 dB SNR an accuracy of 97% on drones and 99% accuracy on non-drones. The accuracy in all our results is normalized by the number of samples in each class, and is thus equivalent to the F1 score. Unlike the first approach, the second approach can easily be updated with new classes without retraining the entire classifier. Such design is helpful in target identification.

In both cases, the accuracy may be improved by including RCS, trajectory and velocity features in the target recognition process. This, as well as extending the results to radars operating with longer dwell times is left for future work.

Acknowledgment

We thank Thomas Lindgren, Thomas Forsström, Andreas Sjöberg, Håkan Edvardsson, Karl-Johan Hällgren, Peter Furustig, Dennis Edblom, Didrik Book, and Oskar Wretstam for their help with collecting radar measurements.

Chapter 7

Low-Angle Target Tracking in Sea Surface Multipath

In this study we use a CNN as a parameter estimator. We estimate the path difference in sea surface multipath, and use this estimate to obtain more accurate estimates of the target height during tracking. The performance of the CNN based estimator is compared with maximum likelihood and least squares methods. The results and methods presented in this chapter are published in paper C; see page xi.

7.1 Introduction

In order to avoid detection, sea-skimming targets fly at very low altitudes, sometimes on the order of meters. Radar systems for tracking such targets will thus be faced with a low angle tracking problem where reflections from the sea surface, or multipath interference, may distort the height estimate, as illustrated in Fig. 7.1. This has been a frequently discussed problem since the early days of radar [4, Sec. 6.11].

Although this type of interference has been of particular concern in naval situations [154], it may arise in most types of terrain [155]. By "low angle" we refer to scenarios when the target elevation angle is close to, or less than, the elevation beamwidth of the radar.

Apart from earlier methods of filtering out the multipath error over time [156], three basic approaches appear in the literature for solving low angle tracking in multipath. The first is to regard the reflection from the sea surface as a second target, and estimate the angles of both the true target and the surface reflection [157]. This is illustrated in Fig. 7.2a.

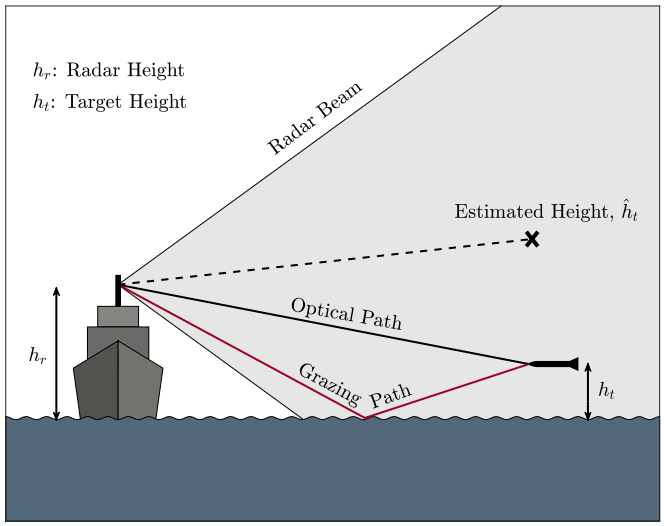
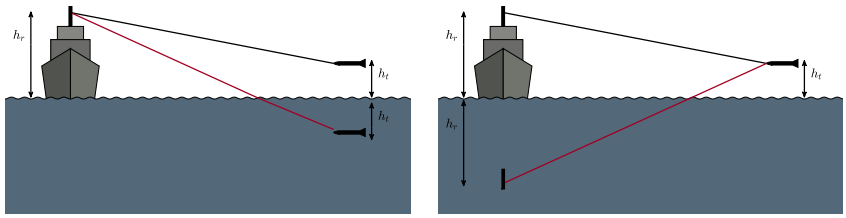


Figure 7.1: Illustration of a low angle scenario. The sea surface is well within the radar beamwidth. Specular reflections from the grazing path interferes with the direct reflection from the optical path, resulting in significant errors in the height estimate.

Methods for angle estimation such as multiple signal classification (MUSIC) [158] and estimation of signal parameters via rotational invariance techniques (ESPRIT) [22] will however perform poorly in these situations due to the strong correlation between the specular and direct reflection. To better handle correlation between signals in the general case, spatial smoothing was introduced in [159], which has the effect of decorrelating signals at the cost of a slightly reduced aperture. The level of decorrelation is limited to the number of antenna elements. See [160–162] for a detailed analysis of this. Other methods such as maximum likelihood (ML) and weighted subspace fitting (WSF)/method of direction estimation (MODE) [163, 164] do however not assume uncorrelated signals.

In [156] it was shown that the complex angle of a monopulse antenna could be used for resolving the direct and specular components since the imaginary part of the monopulse angle contains information about the specular and direct components.

The second approach is to cancel out the multipath component in the received signal. In [165] this is done by placing a null in the direction of the grazing path using adaptive beamforming. This requires high SNRs as it will unavoidably also reduce the signal level in the optical path when these are close. In [166],



(a) (b)

Figure 7.2: Illustration of the two target model (a) and the the interferometric model (b). The true signal path is shown in Fig. 7.1. In (b) the receive antenna is modeled as two elements separated by a distance $2h_r$, where h_r is the radar antenna height above the sea surface.

and [167] neural networks are used to remove the effect of multipath in the sampled covariance matrix. Super resolution methods, such as MUSIC and ESPRIT, can then be used to estimate the angle of arrival of true targets using the modified covariance matrix. In [168] orthogonal frequency-division multiplexing is used to mitigate nulls that occur when the direct and specular reflections cancel each other out.

The third approach uses geometric assumptions about the direct and specular reflections and is thus model based. An estimate of the path difference between the optical and grazing paths, together with the target range and a geometric model, can be used to estimate the target height/angle. This can result in relatively high accuracy when compared to pure direction finding methods that do not consider any geometric models, especially at low SNRs. We will refer to this approach as the interferometric approach since the path difference can be derived from the interference caused by the specular reflection. Note that a path difference in itself gives no information about the true target angle. The interferometric model for a flat sea surface is illustrated in Fig. 7.2b.

The interferometric approach is used in the CelsiusTech high accuracy sea-skimmer estimator (CHASE) algorithm in [169] and [36]. With an amplitude monopulse antenna, CHASE utilises the “complex indicated angle” first presented in [156], and further discussed in [154], to estimate the path difference between the optical and grazing paths. In addition to this, the CHASE algorithm uses frequency diversity, a technique commonly used for reducing sensitivity in multipath with many specular components [170], and for avoiding jamming. One can regard the specular reflection as an echo received at a separate antenna located $2h_r$ below

the actual antenna [169], where h_r is the radar height. This results in a very large synthetic aperture and therefore high accuracy, but with many grating lobes. By illuminating the target at several frequencies the correct lobe can be determined, [36].

In this study we apply the interferometric approach, but treat it as a pattern recognition problem. By illuminating the target at different frequencies, different interference levels are obtained. When plotted, these form distinct patterns. We use this as input to a CNN which we train to estimate the path difference between the optical and grazing paths. This is similar to our study in Chapter 4, where we used frequency agility and a CNN for non-coherent pulse compression. In the case of multipath however, the considered bandwidth is far too small for resolving the direct and specular echoes with the method in Chapter 4.

Neural networks have become popular methods for solving angle of arrival problems in general [171–175]. Once trained, a neural network is relatively cheap to evaluate, as it typically only involves a series of matrix multiplications. Another advantage is that they are data driven, i.e they do not need an explicit model of the data statistics or receiver characteristics. As we have discussed earlier, it is often impractical to collect enough labeled data to train a CNN model on real measurements. Models may therefore still necessary and will be utilised in this study. These models do however not need to be modified or approximated for the sake of analytical convenience. As long as the model can be sampled, a neural network solution can be formed. The question is however, how close will this solution be to the optimal solution?

In this study we sample data from a mathematical model. Whenever this is done, one could always form a LS solution using the model. This may involve an extensive grid search, but it is nonetheless an interesting benchmark. It may also be possible to formulate a ML estimator from the model. Solving these LS and ML problems may however be more computationally demanding than a CNN. We will, as a benchmark, compare the pattern recognition method of a CNN to the pattern matching methods of LS and ML.

The remainder of this chapter is organized as follows. In Section 7.2 we present modeling assumptions and mathematical formulations for the geometry between target and radar, sea surface reflections, radar sensor and the received signal. In Section 7.3 we present the ML and LS based methods for estimating the target height as well as our proposed CNN based solution. The results are shown in Section 7.4 and conclusions as well as directions for future work are given in Section 7.5.

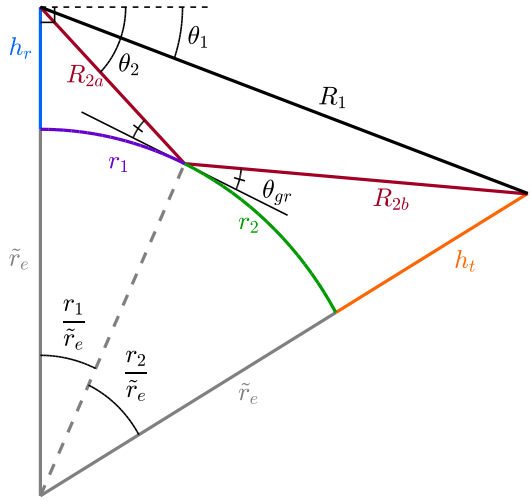


Figure 7.3: Geometry of the scenario in Fig.7.1. The radar is positioned at height h_r and the target at height h_t. The effective earth radius, $\tilde{r}_e$, is the earth radius scaled by a factor k. The top dashed line marks the normal to the antenna and has angle zero. Note that θ_2 is always negative and θ_{gr} is always positive.

7.2 System Model

In this section we present the models and assumptions used in this study. We generate data from this model and use it for training the CNN presented in Section 7.3.3.

7.2.1 Geometric Model

The geometry of the setup is shown in Fig. 7.3. To account for refraction, i.e. the fact that electromagnetic waves propagate along a curved line rather than a straight, we use the effective earth radius model. This allows paths to be treated as straight lines by scaling the earth radius, r_e, by an effective radius factor k. A common value is $k = 4/3$ for altitudes less than 4 km [3, p. 304].

In practice the appropriate value of this factor may vary considerably with weather conditions, particularly at low altitudes. Ducting, a phenomenon which can be thought of as a layer of air acting as a wave guide due to having different temperature and humidity, may considerably change the effective radar horizon and thus the value of k, [176], [177]. The height required for a duct to "trap" a wave is also frequency dependent [178, Sec. 5.2.9]. We will use the standard value of

$k = 4/3$ in this study, and in Section 7.4.5.4 we discuss the error in the height estimate when assuming an incorrect value of k.

Using the law of cosines, we have from the geometry in Fig. 7.3 that

$$r_1 + r_2 = r = \tilde{r}_e \cos^{-1}\left(\frac{(h_r + \tilde{r}_e)^2 + (\tilde{r}_e + h_t)^2 - R_1^2}{2(\tilde{r}_e + h_r)(\tilde{r}_e + h_t)}\right) \tag{7.1}$$

$$R_{2a} = \sqrt{(\tilde{r}_e + h_r)^2 + \tilde{r}_e^2 - 2\tilde{r}_e(\tilde{r}_e + h_r)\cos\left(\frac{r_1}{\tilde{r}_e}\right)} \tag{7.2}$$

$$R_{2b} = \sqrt{(\tilde{r}_e + h_t)^2 + \tilde{r}_e^2 - 2\tilde{r}_e(\tilde{r}_e + h_t)\cos\left(\frac{r_2}{\tilde{r}_e}\right)} \tag{7.3}$$

$$\theta_2 = -\sin^{-1}\left(\frac{2\tilde{r}_e h_r + h_r^2 + R_{2a}^2}{2R_{2a}(\tilde{r}_e + h_r)}\right) \tag{7.4}$$

$$\theta_1 = \theta_2 + \cos^{-1}\left(\frac{R_{2a}^2 + R_1^2 - R_{2b}^2}{2R_{2a}R_1}\right) \tag{7.5}$$

$$\theta_{gr} = -\theta_2 - r_1/\tilde{r}_e \tag{7.6}$$

where

- $\tilde{r}_e = 4/3 r_e$ is the effective earth radius
- h_r is the radar height
- h_t is the target height
- R_1 is the optical path, or target range
- $R_2 = R_{2a} + R_{2b}$ is the grazing path
- $r = r_1 + r_2$ is the ground range
- θ_1 is the angle of the direct reflection
- θ_2 is the angle of the specular reflection
- θ_{gr} is the grazing angle.

The range r_1 is given in [165] as

$$r_1 = \frac{r}{2} - p\sin\left(\frac{\phi}{3}\right) \tag{7.7}$$

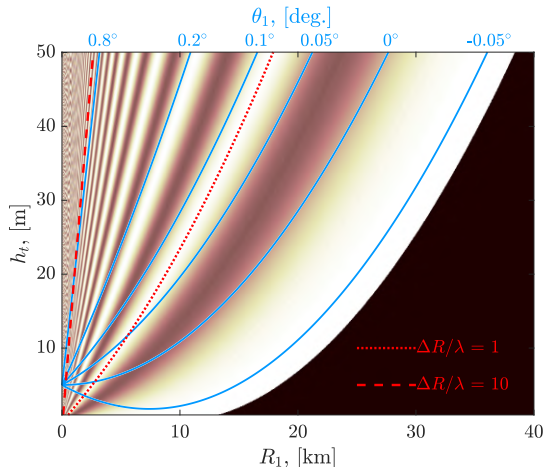


Figure 7.4: Multipath interference pattern. The image shows the multipath interference pattern as a function of target height and range for $h_r = 5$ m and frequency $f_c = 16$ GHz. The magnitude of the interference depends on the strength of the reflection coefficient. A constant value of ΔR gives a specific line along a contour in this image, e.g, the dashed lines which mark lobe one and ten counting from zero. The height can thus be estimated from knowing ΔR and R_1. Note that the target height at 0° elevation is 50 m at 28 km range. This differs significantly from a flat earth assumption where $h_t = h_r = 5$ m. The dark area to the bottom right is beyond the radar horizon.

where

$$p = \frac{1}{\sqrt{3}} \sqrt{4\tilde{r}_e(h_t + h_r) + r^2} \tag{7.8}$$

$$\phi = \sin^{-1}\left(\frac{2\tilde{r}_e(h_t - h_r)r}{p^3}\right). \tag{7.9}$$

Let λ be the wavelength and let the difference between the optical and grazing paths be

$$\Delta R = R_2 - R_1. \tag{7.10}$$

The received echoes will add up constructively when $\Delta R/\lambda$ is an integer and destructively when $2\Delta R/\lambda$ is an odd integer, giving rise to the characteristic periodic patterns in multipath interference. An example of this is shown in Fig. 7.4. The location of the nulls thus depend on ΔR and λ, where λ unlike ΔR can be chosen by the transmitter. By varying λ we aim to estimate ΔR from which we

can obtain an estimate of h_t. If the final tracking is based on θ_1 rather than h_t, the angle can be obtained from h_t using the law of cosines or the approximation in [179, eq. 2.39]

$$\begin{aligned}\theta_1 &= \sin^{-1}\left(\frac{(\tilde{r}_e + h_t)^2 - R_1^2 - (\tilde{r}_e + h_r)^2}{2R_1(\tilde{r}_e + h_r)}\right) \\ &\approx \sin^{-1}\left(\frac{h_t - h_r}{R_1} - \frac{R_1}{2\tilde{r}_e}\right)\end{aligned} \tag{7.11}$$

In this study, we will only consider elevation tracking from the basis of height tracking.

7.2.2 Surface Reflection Model

The total reflection coefficient of the sea surface is modeled as [3, p. 290-296].

$$\Gamma = \rho_s \Gamma_0 D \tag{7.12}$$

where ρ_s accounts for the attenuation caused by the roughness of the sea surface, Γ_0 is the Fresnel reflection coefficient for sea water, and D is a divergence factor that accounts for the attenuation caused by the earths curved surface. The sea surface roughness coefficient ρ_s is modeled according to [3, eq. 6.2.13]

$$\rho_s = e^{-\frac{1}{2}\left(\frac{4\pi\sigma_h}{\lambda}\right)^2 \sin^2(\theta_{gr})} \tag{7.13}$$

where σ_h is the RMS wave height. This is related to the significant wave height $H_{1/3}$ as $4\sigma_h = H_{1/3}$ [179, Sec. 2.11]. For low sea states and low grazing angles, σ_h is close to unity. If e.g. $H_{1/3} < 0.1$ m and $\theta_{gr} < 1^\circ$ then $\rho_s > 0.94$. The Fresnel coefficient for vertical polarisation is [3, eq. 6.2.12]

$$\Gamma_0 = \frac{\varepsilon_c \sin(\theta_{gr}) - \sqrt{\varepsilon_c - \cos^2(\theta_{gr})}}{\varepsilon_c \sin(\theta_{gr}) + \sqrt{\varepsilon_c - \cos^2(\theta_{gr})}} \tag{7.14}$$

where the relative permittivity, ε_c, is

$$\varepsilon_c = \varepsilon_r - j60\lambda\sigma_e. \tag{7.15}$$

We assume that $\varepsilon_r = 60$ and the conductivity $\sigma_e = 15$ mho/m for salt water, [3, Table. 6.4]. When the radar pulse is reflected on a curved surface it diverges somewhat, yielding a loss relative to a flat surface reflection. This divergence factor

Table 7.1: Pulse Repetition Frequency

Range Interval, [km]	**PRF, [kHz]**
0 - 8	16
8 - 16	8
16 - 32	4
32 - 75	2

The values are chosen such that the range is less than the ambiguous range.

is [6, Sec. 5.2]

$$D = \frac{1}{\sqrt{1 + \frac{2r_1 r_2}{\tilde{r}_e r \sin\theta_{gr}}}}. \tag{7.16}$$

7.2.3 Radar Model

We assume a frequency agile pulsed radar with monopulse functionality and vertical polarisation (relevant for (7.14)). The presented method does not require any particular antenna type beyond this. Here we will assume a phase monopulse system with a 64 element phased ULA. The data is sampled in two channels; the first and the second half of the array. We choose center frequency at $f_c = 16$ GHz with random frequency agility within a 1.6 GHz bandwidth (10% of center frequency). We let the PRF vary with range according to Table 7.1. We only consider tracking in elevation and we therefore present no assumptions regarding the radar's azimuth or pulse width specifications. To reduce sidelobe interference we add a Taylor window, see Section 2.8, to the transmit and receive steering vectors. For this we choose a sidelobe suppression level of -40 dB over 10 lobes. The two way 3 dB beamwidth of the ULA is then $1.6°$.

7.2.4 Signal Model

Assuming antenna elements with omnidirectional gain, the total received echo at antenna element i for a ULA with $2M$ elements is

$$\begin{aligned} s_i(t,f) \propto \text{Real}\Bigg[\sum_{m=1}^{2M} & p(t\beta_{im1} - \tau_{im1}) \frac{\alpha_m \sqrt{\sigma}}{R_1^2} e^{j2\pi f(\beta_{im1} t - \tau_{im1})} \\ & + p(t\beta_{im2} - \tau_{im2}) \frac{\alpha_m \Gamma \sqrt{\sigma}}{R_1 R_2} e^{j2\pi f(\beta_{im2} t - \tau_{im2})} \\ & + p(t\beta_{im3} - \tau_{im3}) \frac{\alpha_m \Gamma \sqrt{\sigma}}{R_2 R_1} e^{j2\pi f(\beta_{im3} t - \tau_{im3})} \\ & + p(t\beta_{im4} - \tau_{im4}) \frac{\alpha_m \Gamma^2 \sqrt{\sigma}}{R_2^2} e^{j2\pi f(\beta_{im4} t - \tau_{im4})} \Bigg] \end{aligned} \tag{7.17}$$

where

- $p(t)$ is the pulse envelope
- β_{imr} is the Doppler time scaling factor between element i and m for the four possible reflection paths r. These are $R_1 + R_1$ ($r = 1$), $R_1 + R_2$ ($r = 2$), $R_2 + R_1$ ($r = 3$) and $R_2 + R_2$ ($r = 4$)
- τ_{imr} is the time delay for path r from element i to m
- σ is the RCS
- α_m is a complex valued gain at element m
- f is the frequency.

Here we have assumed that the grazing angle is the same for all elements, i.e. Γ is the same for all m. Using the far-field approximation, the time delay between antenna elements i and m for each path is

$$\tau_{im1} \approx \frac{R_1 + R_1 - d_i \sin(\theta_1) - d_m \sin(\theta_1)}{c} \tag{7.18}$$

$$\tau_{im2} \approx \frac{R_1 + R_2 - d_i \sin(\theta_1) - d_m \sin(\theta_2)}{c} \tag{7.19}$$

$$\tau_{im3} \approx \frac{R_2 + R_1 - d_i \sin(\theta_2) - d_m \sin(\theta_1)}{c} \tag{7.20}$$

$$\tau_{im4} \approx \frac{R_2 + R_2 - d_i \sin(\theta_2) - d_m \sin(\theta_2)}{c}. \tag{7.21}$$

The antenna element positions are

$$d_m = \left(\frac{2M+1}{2} - m \right) \frac{\lambda_c}{2} \tag{7.22}$$

for $m = 1, ..., 2M$, and λ_c is the wavelength of the center frequency. Let the complex valued transmit gain, or steering vector, at each element be

$$\alpha_m = w_m e^{-j2\pi d_m \sin(\theta_{Tx})/\lambda} \tag{7.23}$$

where w_m is a window weight and θ_{Tx} is the desired transmit direction. After complex downconversion with frequency f_0, and assuming that

- the radial velocity between the target and radar, v_r, is the same for all paths, i.e. $\beta_{imr} = \beta \approx 1 + \frac{2v_r}{c}$, with $v_r > 0$ when the target is moving towards the radar
- the pulse length is long enough such that $p(t\beta_{imr} - \tau_{imr}) \approx p(t - \tau_1) = p(t - 2R_1/c)$ for all i, m and r
- $R_1^2 \approx R_1 R_2 \approx R_2^2$

we have from (7.10), (7.17), (7.18) - (7.21) and (7.23)

$$\begin{aligned} s_i(t, \tilde{f}) \propto p(t - \tau_1) \frac{\sqrt{\sigma}}{R_1^2} e^{j2\pi \left(f - f_0 + \frac{2v_r \tilde{f}}{\lambda_c} \right) t} e^{-j4\pi R_1 \tilde{f}/\lambda_c} \\ \Big[e^{j2\pi d_i \sin(\theta_1)\tilde{f}/\lambda_c} \left(G(\theta_1) + G^*(\theta_1) \right) \\ + e^{j2\pi d_i \sin(\theta_1)\tilde{f}/\lambda_c} \left(G(\theta_2) + G^*(\theta_2) \right) \Gamma e^{-j2\pi \Delta R \tilde{f}/\lambda_c} \\ + e^{j2\pi d_i \sin(\theta_2)\tilde{f}/\lambda_c} \left(G(\theta_1) + G^*(\theta_1) \right) \Gamma e^{-j2\pi \Delta R \tilde{f}/\lambda_c} \\ + e^{j2\pi d_i \sin(\theta_2)\tilde{f}/\lambda_c} \left(G(\theta_2) + G^*(\theta_2) \right) \Gamma^2 e^{-j4\pi \Delta R \tilde{f}/\lambda_c} \Big] \end{aligned} \tag{7.24}$$

where $\tilde{f} = f/f_c$ is a normalized frequency and

$$G(\theta_x) = \sum_{m=1}^{M} w_m e^{j2\pi d_m [\sin(\theta_x) - \sin(\theta_{Tx})] \tilde{f}/\lambda_c}. \tag{7.25}$$

Note that $G(\cdot)$ is a summation over half the array. In single target tracking, we sample $s_i(t, \tilde{f})$ approximately at the target, i.e. $t \approx \tau_1$. We do this in two channels

where the receive steering vector is the same as the transmit steering vector

$$x_1(\tilde{f}) = \sum_{i=1}^{M} \alpha_i s_i(\tau_1, \tilde{f}) + u_1 \tag{7.26}$$

$$x_2(\tilde{f}) = \sum_{i=M+1}^{2M} \alpha_i s_i(\tau_1, \tilde{f}) + u_2 \tag{7.27}$$

where $u_1 \sim \mathcal{CN}\left(0, \sigma_u^2\right)$ and $u_2 \sim \mathcal{CN}\left(0, \sigma_u^2\right)$ are independent complex Gaussian noise components. We obtain the phase monopulse sum channel from (7.23) - (7.27) as

$$\begin{aligned}
\Sigma_x(\tilde{f}) =& x_1(\tilde{f}) + x_2(\tilde{f}) = \sum_{i=1}^{2M} \alpha_i s_i(\tau_1, \tilde{f}) + u_1 + u_2 \\
=& 4C\frac{\sqrt{\sigma}}{R_1^2} e^{j\phi} G_I^2(\theta_1) \\
& \left[1 + 2\delta e^{-j2\pi\tilde{f}\Delta R/\lambda_c} + \delta^2 e^{-j4\pi\Delta R/\lambda}\right] + u_1 + u_2 \\
=& 4C\frac{\sqrt{\sigma}}{R_1^2} e^{j\phi} G_I^2(\theta_1) F^2(\delta_1) + u_1 + u_2
\end{aligned} \tag{7.28}$$

where $G_I(\cdot)$ denotes the in-phase, or real part of (7.25) and

$$e^{j\phi} = e^{j2\pi\left(f - f_0 + \frac{2v_r\tilde{f}}{\lambda_c}\right)\tau_1} e^{-j4\pi R_1\tilde{f}/\lambda_c} \tag{7.29}$$

$$\begin{aligned}
F^2(\delta) =& 1 + 2\delta e^{-j2\pi\Delta R\tilde{f}/\lambda_c} + \delta^2 e^{-j4\pi\Delta R\tilde{f}/\lambda_c} \\
=& \left(1 + \delta e^{-j2\pi\Delta R\tilde{f}/\lambda_c}\right)^2
\end{aligned} \tag{7.30}$$

$$\delta_1 = \Gamma\frac{G_I(\theta_2)}{G_I(\theta_1)}. \tag{7.31}$$

Note that we have replaced the proportionality sign with a complex valued constant C. The difference channel is

$$\begin{aligned}
\Delta_x(\tilde{f}) =& x_1(\tilde{f}) - x_2(\tilde{f}) \\
=& j4C\frac{\sqrt{\sigma}}{R_1^2} e^{j\phi} G_I(\theta_1) G_Q(\theta_1) F(\delta_1) F(\delta_2) + u_1 - u_2
\end{aligned}$$

where $G_Q(\cdot)$ denotes the quadrature component, or imaginary part of (7.25), and

$$\delta_2 = \Gamma \frac{G_Q(\theta_2)}{G_Q(\theta_1)}. \tag{7.32}$$

The phase comparison monopulse ratio for the elevation angle is

$$\begin{aligned}\eta(\tilde{f}) = \frac{j\Delta_x(\tilde{f})}{\Sigma_x(\tilde{f})} &= -\frac{\tilde{C}G_I(\theta_1)G_Q(\theta_1)F(\delta_1)F(\delta_2) + u_1 - u_2}{\tilde{C}G_I^2(\theta_1)F^2(\delta_1) + u_1 + u_2} \\ &= -\frac{G_Q(\theta_1)F(\delta_2) + u_{\mathrm{n}}}{G_I(\theta_1)F(\delta_1) + u_{\mathrm{d}}}\end{aligned} \tag{7.33}$$

where

$$u_{\mathrm{n}} = \frac{u_1 - u_2}{\tilde{C}G_I(\theta_1)F(\delta_1)} \tag{7.34}$$

$$u_{\mathrm{d}} = \frac{u_1 + u_2}{\tilde{C}G_I(\theta_1)F(\delta_1)} \tag{7.35}$$

$$\tilde{C} = 4|C|\frac{\sqrt{\sigma}}{R_1^2}e^{j(\phi + \angle C)}. \tag{7.36}$$

We note that if $\Gamma = 0$ then $F(\delta_1) = F(\delta_2) = 1$, and (7.33) becomes an undistorted phase monopulse ratio. The monopulse slope, κ with no multipath, i.e. $\Gamma = 0$, can be calculated by taking the derivative of (7.33) with respect to θ_1 and evaluating at $\theta_{Tx} = \theta_1$. With a symmetric window and ignoring the noise terms, the phase monopulse slope is

$$\kappa(\tilde{f}) = -\frac{1}{\tilde{f}\pi\cos(\theta_{Tx})}\frac{\sum_{m=1}^{M} w_m}{\sum_{m=1}^{M}(M + 0.5 - m)w_m} \tag{7.37}$$

from which the phase monopulse angle estimate of θ_1 is obtained as

$$\hat{\theta}_1^{\mathrm{MP}} = \theta_{Tx} + \varepsilon_{\theta_1} \tag{7.38}$$

where

$$\varepsilon_{\theta_1} = \kappa(\tilde{f})\mathrm{Real}\left[\eta(\tilde{f})\right]. \tag{7.39}$$

This estimate will be quite accurate within the 3 dB beamwidth of the antenna provided sufficient SNR and no interference from other targets or, as is the case in this study, multipath reflections. Multipath interference will cause a slight bias in the angle estimate, which will have a periodic pattern. We can see this by noting

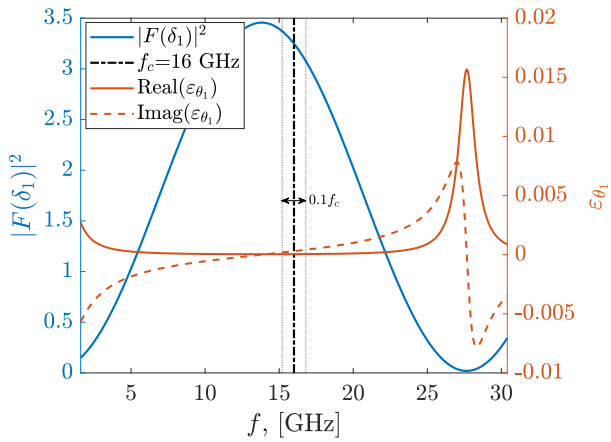


Figure 7.5: The real and imaginary part of the angle error ε_{θ_1} and the term $|F(\delta_1)|^2$ in (7.40) as a function of frequency. In this case $h_r = 5$ m, $h_t = 15$ m, $R_1 = 10$ km and $H_{1/3} = 0.1$ m. Estimating the frequency of $|F(\delta_1)|^2$, i.e. a frequency over frequency, in a traditional manner such as Fourier analysis will be difficult even with the relatively large bandwidth of 1.6 GHz. We note that the same periodicity appears in the angle error ε_{θ_1}.

that $F(\delta_1)$ in (7.33) has magnitude

$$|F(\delta_1)|^2 = \left|1 + \delta_1 e^{-j2\pi\Delta R\tilde{f}/\lambda_c}\right|^2 = 1 + |\delta_1|^2 + 2|\delta_1|\cos(2\pi\Delta R\tilde{f}/\lambda_c + \angle\delta_1). \quad (7.40)$$

The magnitude of the sum channel, (7.28), will thus vary as a squared cosine as λ changes and/or as ΔR changes. The phase monopulse ratio in (7.33) will not be sinusoidal over λ, but will have the same periodicity. Over a sufficiently small number of pulses, R_1 and h_t can be assumed constant. Estimating ΔR can then be formulated as a frequency estimation problem over frequency, where the bandwidth over all pulses will determine the "signal length." This is illustrated in Fig. 7.5, where $|F(\delta_1)|^2$ is plotted for a specific scenario. We see that even a relatively large bandwidth of 1.6 GHz only captures a fraction of the period, making the frequency, i.e. $\Delta R/\lambda_c$, difficult to estimate. To have at least one period within the bandwidth of $0.1f_c$, the target must be in the area above the dashed line that marks $\Delta R/\lambda = 10$ in Fig. 7.4. This rules out most of the interesting sea skimming scenarios when $h_t <$ 50 m.

7.3 Parameter Estimation

We will look at four different methods for estimating the target height h_t; a CNN based approach, two LS approaches and one based on ML. All these methods fall under the interferometric approach discussed in Section 7.1. In the LS and ML methods we aim to match the data to a model via a parameter search. With a neural network, we instead aim to form an inverse function of the model to directly obtain the parameter estimates.

A neural network will therefore offer an advantage whenever a multi dimensional search/optimisation becomes too computationally demanding and/or when analytical formulations of the model are intractable.

7.3.1 Height Estimation

As discussed in Section 7.1, the target height can be estimated from the difference between the grazing path and optical path. For a flat earth approximation, the grazing path R_2 is

$$R_2 \approx \sqrt{(h_t + h_r)^2 + r^2}. \tag{7.41}$$

where the ground path for a flat earth approximation is

$$r \approx \sqrt{R_1^2 - (h_t - h_r)^2}. \tag{7.42}$$

Using (7.41) and (7.42) in (7.10), and the Taylor expansion $\sqrt{1+x} \approx 1 + x/2$ for small x, gives us the difference between the grazing and optical path in range

$$\Delta R = R_2 - R_1 \approx R_1 \sqrt{1 + \frac{4h_t h_r}{R_1^2}} - R_1 \approx \frac{2h_t h_r}{R_1}, \tag{7.43}$$

and we could thus obtain h_t from ΔR, R_1 and h_r, had the earth been flat. To account for the earth's curvature, a bias term needs to be added to (7.43). This is further discussed in [6, Sec. 2.13]. As there is no simple analytical expression for this term, we will resort to the relatively simple solution of interpolating h_t from a table which we have sampled using (7.1)-(7.10) for a constant h_r. Specifically, we use bilinear interpolation on ΔR and R_1 to obtain h_t, instead of the approximation in (7.43).

There are some interesting observations to be made regarding the choice of h_r. Using again the flat earth approximation, the difference between the grazing and

optical path in angle is

$$\theta_1 - \theta_2 = \Delta\theta \approx \tan^{-1}\left(\frac{h_t - h_r}{r}\right) - \tan^{-1}\left(\frac{-(h_t + h_r)}{r}\right). \tag{7.44}$$

A first order Taylor expansion on $\tan^{-1}(x)$ for small x, and inserting (7.42) in (7.44) gives us

$$\Delta\theta \approx \frac{2h_t}{R_1\sqrt{1 - \frac{(h_t - h_r)^2}{R_1^2}}} \approx \frac{2h_t}{R_1}. \tag{7.45}$$

We see that the effect of the radar height h_r will be negligible in methods that rely only on angle separation. In contrast to this, we see in (7.43) that ΔR is proportional to h_r. The choice of h_r is thereby of direct relevance when estimating h_t from ΔR. In Section 7.4.4 we further compare the direction finding method with the interferometric method.

7.3.2 Maximum Likelihood and Least Squares

We assume that the coherence is lost between frequency shifts, i.e. the phase of C in (7.28) changes randomly between pulses with different frequencies. Estimating ΔR directly from $x_1(\tilde{f})$ and $x_2(\tilde{f})$ in (7.26)-(7.27) would then involve a new unknown phase for each pulse. We can overcome this by instead estimating ΔR from the phase monopulse ratio where the phase of C only appears in the noise terms as a random complex rotation of an already random number (see (7.33)-(7.35)).

An alternative is to take the modulus of (7.28). For sufficiently high SNRs this approximately removes the phase component of C. We will look at ML on the phase monopulse ratio $\eta(\tilde{f})$, LS on $\eta(\tilde{f})$ and LS on the modulus of (7.28), $|\Sigma_x(\tilde{f})|$. LS is an attractive solution since it does not require any knowledge of the noise distribution and often performs well in practice.

7.3.2 ML on $\eta(\tilde{f})$

The phase monopulse ratio in (7.33) is a ratio between two complex Gaussian random variables. They are seemingly dependent since both u_1 and u_2 appear in the numerator and denominator. They are however uncorrelated (and, hence, independent since they are Gaussian) if the variances in u_1 and u_2 are equal, i.e.

$$\begin{aligned}&\mathbb{E}\left[(u_1 - u_2)(u_1 + u_2)^*\right] \\ &= \mathbb{E}\left[u_1 u_1^*\right] - \mathbb{E}\left[u_2 u_2^*\right] = \sigma_u^2 - \sigma_u^2 = 0.\end{aligned} \tag{7.46}$$

The joint PDF of the real and imaginary parts of such a ratio is given in [180]. After some algebraic rearranging this can written as

$$p(\eta(\tilde{f})|\boldsymbol{\beta}_{\text{ML}}) = \frac{\exp\left(-\gamma' \frac{|\eta-\rho_\eta|^2}{|\eta|^2+1}\right)}{\pi\left(|\eta|^2+1\right)^2}\left(1+\gamma' \frac{|\eta^*\rho_\eta+1|^2}{|\eta|^2+1}\right) \tag{7.47}$$

where

$$\rho_\eta = -\frac{G_Q(\theta_1)F(\delta_2)}{G_I(\theta_1)F(\delta_1)} \tag{7.48}$$

$$\begin{aligned}\gamma' &= \frac{1}{2\sigma_u^2} 16|C|^2 \frac{\sigma}{R_1^4} G_I^4(\theta_1)|F(\delta_1)|^4 \\ &= \frac{|\Sigma_x|^2}{2\sigma_u^2}.\end{aligned} \tag{7.49}$$

With independent noise between N_f frequency samples, we find the ML solution by maximizing the log-likelihood of $\eta(\tilde{f}_1), .., \eta(\tilde{f}_{N_f})$, over the unknown parameters in $\boldsymbol{\beta}_{\text{ML}}$ i.e.

$$\boldsymbol{\beta}^\star_{\text{ML}} = \underset{\boldsymbol{\beta}_{\text{ML}}}{\text{argmax}} \sum_{i=1}^{N_f} \log(p(\eta(\tilde{f}_i)|\boldsymbol{\beta}_{\text{ML}})). \tag{7.50}$$

The unknown parameters in $\boldsymbol{\beta}_{\text{ML}}$ are θ_1, θ_2, ΔR, Γ and $\tilde{\sigma} = \sigma/\sigma_u^2$ (or $\tilde{\sigma} = \sigma$ if the noise variance σ_u^2 is known). However, under the geometry assumption in Section 7.2.1, ΔR, θ_1 and θ_2 can all be derived from h_t (and the range R_1). We may also assume that $|\Gamma|$ is constant over frequency and that the argument of Γ is approximately π, see discussions in Section 7.2.2. We can thereby reduce the optimization to three dimensions, i.e.

$$\boldsymbol{\beta}_{\text{ML}} = [h_t, |\Gamma|, \tilde{\sigma}]. \tag{7.51}$$

Even though $\tilde{\sigma}$ can be estimated directly from the sum channel, small errors in this estimate may yield a solution that is much worse than the simple least squares. A joint optimisation over $\tilde{\sigma}$, $|\Gamma|$ and h_t is therefore necessary.

7.3.2 LS on $\eta(\tilde{f})$

We see in (7.47) that for large γ', the likelihood decreases quickly when η deviates from ρ_η. This motivates the least squares solution

$$\boldsymbol{\beta}_{\text{LS}}^{\star} = \underset{\boldsymbol{\beta}_{\text{LS}}}{\text{argmin}} \sum_{i=1}^{N_f} \left| \eta(\tilde{f}_i) - \rho_\eta(\tilde{f}_i) \right|^2 \tag{7.52}$$

$$\boldsymbol{\beta}_{\text{LS}} = [h_t, |\Gamma|] \tag{7.53}$$

which is computationally cheaper than the ML of (7.50). This requires an optimization in two dimensions, h_t and $|\Gamma|$ under the same assumptions as in the ML case.

7.3.2 LS on $|\Sigma_x(\tilde{f})|$

Let

$$z = |\Sigma_x(\tilde{f})|. \tag{7.54}$$

Since $\Sigma_x(\tilde{f}) \sim \mathcal{CN}\left(\mu_\Sigma, 2\sigma_u^2\right)$ where

$$\mu_\Sigma = 4C\frac{\sqrt{\sigma}}{R_1^2} e^{j\phi} G_I^2(\theta_1) F^2(\delta_1), \tag{7.55}$$

z will follow a Rice distribution with PDF

$$p(z|\rho_z, \sigma_u) = \frac{z}{\sigma_n^2} \exp\left(-\frac{\left(z^2 + \rho_z^2\right)}{2\sigma_n^2} \right) I_0\left(\frac{z\rho_z}{\sigma_n^2} \right) \tag{7.56}$$

$$\rho_z = |\mu_\Sigma| = 4|C|\frac{\sqrt{\sigma}}{R_1^2} |G_I(\theta_1)|^2 |F(\delta_1)|^2 \tag{7.57}$$

where $I_0(\cdot)$ is the modified Bessel function of the first kind with order zero. The maximum likelihood of this involves a search over at least three parameters, h_t, $|\Gamma|$ and σ. It also involves the evaluation of the Bessel function, making it a computationally heavy solution. There are approximations for the Bessel function [181], but these are more valid for high SNRs, i.e. $\rho_z^2/\sigma_u^2 \gg 1$, at which point the Rice distribution is well approximated by the Gaussian distribution. We will therefore resort to the least squares solution. The asymptotic mean of z is in such cases ρ_z and is, as desired, independent of the argument of C. Using (7.40) this can be written as

$$\rho_z = g_0 \left(1 + |\delta|^2 + 2|\delta_1| \cos(2\pi\Delta R\tilde{f}/\lambda_c + \angle\delta_1) \right) \tag{7.58}$$

where

$$g_0 = 4|C|\frac{\sqrt{\sigma}}{R_1^2}|G_I(\theta_1)|^2 \tag{7.59}$$

or put in matrix form with all N_f samples

$$\boldsymbol{\rho}_z = \boldsymbol{A}\boldsymbol{q} \in \mathbb{R}^{N_f \times 1} \tag{7.60}$$

where

$$\boldsymbol{A} = [\mathbf{1}, \boldsymbol{a}] \in \mathbb{R}^{N_f \times 2} \tag{7.61}$$

$$\boldsymbol{a} = \cos(2\pi\Delta R\tilde{\boldsymbol{f}}/\lambda_c + \angle\delta_1) \tag{7.62}$$

$$\boldsymbol{q} = g_0 \begin{bmatrix} 1 + |\delta_1|^2 \\ 2|\delta_1| \end{bmatrix} \tag{7.63}$$

$$\tilde{\boldsymbol{f}} = [\tilde{f}_1, ..., \tilde{f}_{N_f}]^T \tag{7.64}$$

$$\boldsymbol{\rho}_z = [\rho_{z_1}, ..., \rho_{z_{N_f}}]^T. \tag{7.65}$$

We again assume that $\angle\delta_1 = \angle\Gamma \approx \pi$. If we also assume that $\boldsymbol{q}$ is constant over frequency, the problem reduces to a one-dimensional search over ΔR

$$\Delta R^\star = \underset{\Delta R}{\operatorname{argmin}} \, ||\boldsymbol{z} - \boldsymbol{A}\boldsymbol{q}'||_2^2 \tag{7.66}$$

where $\boldsymbol{q}'$ is, for each ΔR, obtained by solving the non-negative least squares problem:

$$\boldsymbol{q}' = \underset{\boldsymbol{q}}{\operatorname{argmin}} \, ||\boldsymbol{z} - \boldsymbol{A}\boldsymbol{q}||_2^2 \tag{7.67}$$

subject to

$$\boldsymbol{q} \geq 0 \tag{7.68}$$

and

$$\boldsymbol{z} = [|\Sigma_x(\tilde{f}_1)|, ..., |\Sigma_x(\tilde{f}_{N_f})|]^T. \tag{7.69}$$

This can formally be solved via quadratic programming. A computationally cheaper solution is obtained from the linear least squares solution as

$$\boldsymbol{q}' = \max\left(0, (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{z}\right). \tag{7.70}$$

This is a computationally appealing solution, and one that can be applied to any single channel frequency agile radar. It does however suffer from both simplifying assumptions and loss of information as it only utilizes the modulus of the sum channel.

7.3.3 Convolutional Neural Network

In our neural network approach we train a CNN to map $|\Sigma_x(\tilde{f}_i)|^2$ and $\eta(\tilde{f}_i)$ for $i = 1, ..., N_f$ to ΔR without any iterative search. The procedure is as follows.

7.3.3 Data Preprocessing

The input to the estimator is a batch of N_f samples of $|\Sigma_x(\tilde{f})|^2$ and $\eta(\tilde{f})$. These are mapped to the input to the CNN, $\boldsymbol{X} \in \mathbb{R}^{3\times 16}$. The three rows are for $|\Sigma_x(\tilde{f})|^2$, and the real and imaginary parts of $\eta(\tilde{f})$. Both real and imaginary parts of $\eta(\tilde{f})$ are scaled by $\kappa(\tilde{f})$ in (7.37) to remove the frequency and angular dependency in the antenna diagram.

The 16 columns map to equidistant frequency bands in the normalized range [0.95, 1.05]. The N_f samples in a batch are first sorted to the 16 frequency bins. Each column is then normalized by the number of samples in the frequency bin. All elements in the row containing $|\Sigma_x(\tilde{f})|^2$ are also normalized by the mean of $|\Sigma_x(\tilde{f})|^2$ over all N_f samples. No further normalization is needed for $\eta(\tilde{f})$ since it is already normalized.

7.3.3 Data Sampling

We sample data from the model in Section 7.2. The values of the parameters are shown in Table 7.2. The parameters R_1 and h_t are initially sampled uniformly. All samples beyond the radar horizon are omitted, see Fig.7.4. From the remaining samples, N_{tr} training, N_{va} validation, and N_{te} test samples are randomly chosen. Noise was added to the test samples during evaluation to obtain specific SNRs, whereas the SNR for the training and validation sets was randomly selected for each sample. We define the SNR as the SNR for the multipath free scenario, i.e.

$$\gamma = \frac{g_0^2}{2\sigma_u^2} = \frac{16|C|^2|G_I(\theta_1)|^4}{2\sigma_u^2}\frac{\sigma}{R_1^4} \leq \frac{|C|^2}{2\sigma_u^2}\frac{\sigma}{R_1^4} \tag{7.71}$$

where the inequality becomes equality if $\theta_1 = \theta_{Tx}$. This differs from the momentary SNR γ' in (7.49) by the factor $|F(\delta_1)|^4$. The momentary SNR will fluctuate around the specified SNR with minimum gain of zero and maximum gain of 16 (+12 dB), if $|\delta_1| = 1$. See (7.40).

Table 7.2: Parameters for Data Generation

Parameter	**Value**
R_1	Initially uniform in [1, 50] km
h_t	Initially uniform in [0, 500] m
h_r	5 m
$H_{1/3}$	10^q with q uniform in [-2, 0.6] m
N_f	50
f_c	16 GHz
$\tilde{f}$	Uniform in [0.95, 1.05]
v_r	Uniform in [100, 1000] m/s
SNR	Uniform in log scale, [3, 40] dB
N_{tr}	200000
N_{va}	20000
N_{te}	1000

Since multipath is primarily a problem in low sea states, we sampled the significant wave height $H_{1/3}$ in logarithmic scale in the interval [0.01, 4], or sea states 0 to 5. We thereby put more weight to low sea state scenarios.

7.3.3 Labels

As in [36], we normalize ΔR by λ_c

$$\frac{\Delta R}{\lambda_c} = \Delta N = n + \Delta n \tag{7.72}$$

where

$$n = \lfloor \Delta N \rfloor \tag{7.73}$$

$$\Delta n = \Delta N \bmod 1. \tag{7.74}$$

The reasoning for this partitioning is as follows. The multipath interference pattern in Fig. 7.4 can be regarded as a lobe pattern, where the first lobe corresponds to $n = 0$ and so forth. Estimating n is, as we will see, more difficult than estimating Δn. From (7.43) and (7.72) we have that the difference between two consecutive

n in height is approximately

$$h_t(n+1) - h_t(n) \approx \frac{(n+1+\Delta n)\lambda_c R_1}{2h_r} - \frac{(n+\Delta n)\lambda_c R_1}{2h_r} = \frac{\lambda_c R_1}{2h_r} \tag{7.75}$$

which for large R_1 can be significant. An incorrect estimate of n will thus appear as a sudden jump or drop in height. As long as the true n is predicted sufficiently often, these sudden jumps can be filtered out, and errors in n can thus be significantly reduced. This is further discussed in Section 7.3.4.

We form a categorical estimate of n and a continuous but constrained estimate of $\Delta n \in [0, 1]$. We can regard Δn as an angle, where Δn=0.01 and Δn=0.99 are just as close as the two points Δn=0.49 and Δn=0.51. This discontinuity when Δn is close to 0 or 1 will cause ambiguous training. To get around this, we represent Δn as the phase of a two dimensional vector with in-phase and quadrature components

$$\boldsymbol{y}_1 = [\Delta n_I, \Delta n_Q]^T = [\cos(2\pi\Delta n), \sin(2\pi\Delta n)]^T. \tag{7.76}$$

An initial estimate of Δn is then obtained as

$$\Delta\hat{n} = -\frac{\angle(\Delta\hat{n}_I + j\Delta\hat{n}_Q)}{2\pi} \tag{7.77}$$

where $\hat{\boldsymbol{y}}_1 = [\Delta\hat{n}_I, \Delta\hat{n}_Q]^T$ is a CNN output. This estimate is then filtered to obtain a final estimate of Δn.

The label for n is a categorical one hot vector which we denote $\boldsymbol{y}_2$. In addition to $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, we also desire an indicator function for when no multipath is present. If the magnitude of δ_1 in (7.31) is small, which happens when the elevation angle becomes large and/or when the sea state is high, the amplitude of the sinusoid in (7.40) decreases and (7.40) eventually becomes indistinguishable from cases when ΔR is close to zero, i.e. h_t is close to zero. In such cases it is better to estimate h_t from the phase monopulse angle in (7.38). We refer to this as *normal tracking* and let *multipath tracking* denote the tracking based on ΔR. The term $|\delta_1|$ is an intuitive metric for switching between the two tracking methods.

To find a good decision boundary we look at the RMSE between θ_1 and the phase monopulse estimate of θ_1 in (7.38), as a function of $|\delta_1|$. Specially we look at the RMSE of $\hat{\theta}_1^{\text{MP}}$ when $|\delta_1|$ is less than some threshold T_{δ_1}. This is shown in Fig. 7.6. A low value of T_{δ_1} will cause ambiguous training since many samples will then look alike, regardless of ΔR. It is on the other hand desirable since it reduces the RMSE in normal tracking. Conversely, a high value of T_{δ_1} will result in easier training and better accuracy in multipath tracking. The point where $T_{\delta_1} = 0.05$

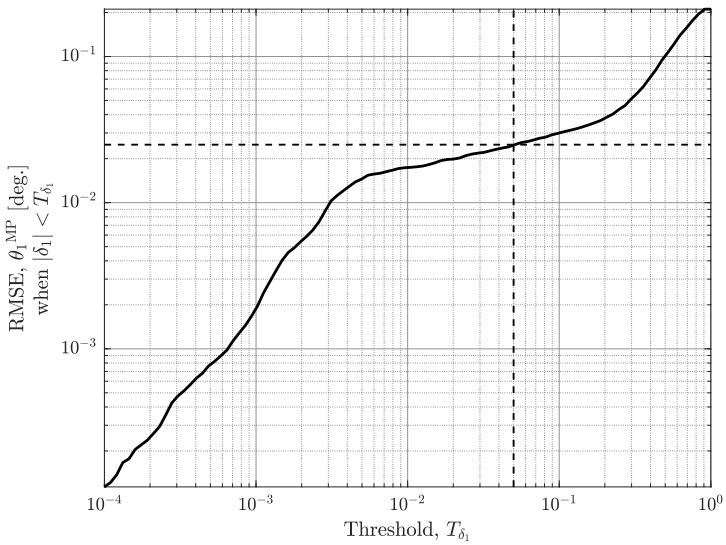


Figure 7.6: The RMSE of $\hat{\theta}_1^{\text{MP}}$ when $|\delta_1| < T_{\delta_1}$, as a function of the threshold T_{δ_1}. The plot is obtained from data sampled according to Table 7.2 but without noise. A natural region for a threshold appears to be $0.008 < T_{\delta_1} < 0.17$. We therefor select the point marked with dashed lines, where $T_{\delta_1} = 0.05$.

in Fig. 7.6 appears to be a good trade-off since the RMSE increases and decreases dramatically above and below this point. We therefore select $T_{\delta_1} = 0.05$ as a threshold between normal and multipath tracking.

The scalar output $\hat{y}_3 \in [0, 1]$, serves as indicator for normal and multipath tracking. We do normal tracking if $\hat{y}_3 > T_{y_3}$ and multipath tracking otherwise. An intuitive value for the threshold T_{y_3} is 0.5 but, as will be discussed later, a higher threshold may be required. This output has label $y_3 = 1$ for samples where $|\delta_1| < T_{\delta_1}$ and $y_3 = 0$ otherwise.

The length of $\boldsymbol{y}_2$, depends on the maximum value of n, n_{max}. This is theoretically infinite, however in practice the limited beamwidth of the radar will limit n_{max} since we use normal tracking when θ_2 is sufficiently separated in angle from θ_1, i.e. when $\hat{y}_3$ is high enough. Considering only samples where $|\delta_1| > T_{\delta_1}$, we have $n \leq 12$ for all samples in the training set. We therefore let $\boldsymbol{y}_2 \in \mathbb{R}^{13\times 1}$ for classes $n = 0, 1, ..., 12$.

7.3.3 Network Architecture

The CNN consists of four blocks. The first, CNN^1, transforms the input $\boldsymbol{X}$ to a 32 element vector, $\boldsymbol{y}_0$. This vector acts as input to three different MLPs.

The first, MLP^1, maps $\boldsymbol{y}_0$ to $\boldsymbol{y}_1$ via two tanh activations since both $\Delta\hat{n}_I$ and $\Delta\hat{n}_Q$ are bounded to the interval [-1, 1]. The second, MLP^2, maps $\boldsymbol{y}_0$ to $\boldsymbol{y}_2$ via a softmax activation. The third, MLP^3, maps $\boldsymbol{y}_0$ to y_3 via a sigmoid activation. The total CNN is illustrated in Fig. 7.7 (red dashed box), and the details of each block are shown in Table 7.3.

The choice of a convolutional architecture came from the observation that the interference pattern is periodic. A CNN was assumed to better recognize these patterns since many examples are translations of other examples. This turned out to indeed be the case when compared to fully dense architectures.

The resulting network is relatively small with only 9488 parameters. The total number of multiplications for evaluating this network is 15232. We can compare this to the LS solution presented in Section 7.3.2.2. Assuming two multiplications for each frequency sample when calculating the magnitude in (7.52); and $2M$ multiplications for calculating each $\rho_\eta(\tilde{f}_i)$ in (7.48) from the $2M$ element array, we have $15232/4MN_f$ multiplications left for the optimization search over h_t and $|\Gamma|$ when compared to the CNN. Note that this does not account for calculating $\hat{\theta}_1$, $\hat{\theta}_2$, and $\Delta\hat{R}$ from a candidate $\hat{h}_t$. This is in our case $15232/6400 \approx 2$. Depending on the chosen optimisation method, this number may be insufficient by several orders of magnitude. The CNN is therefor a significantly cheaper solution in terms of multiplications.

7.3.3 Training the CNN

The network was trained using stochastic gradient descent and Adam optimisation, as described in Section 3.3. The loss for MLP^1 and MLP^3, L_1 and L_3 respectively, was the empirical weighted mean squared error loss and the loss for MLP^2, L_2,

Table 7.3: Neural Network Architecture

CNN[1]				
Input: $\boldsymbol{X}$, Output: $\boldsymbol{y}_0$				
Layer Type	Weight Shape	Strides	Output Shape	Activation
Convolutional	F: 32, K: 3×6	[1,2]	1×6×32	ReLu
Convolutional	F: 32, K: 1×3	[1,3]	1×2×32	ReLu
Fully Connected	32 × 64	-	32×1	ReLu
MLP[1]				
Input: $\boldsymbol{y}_0$, Output: $\boldsymbol{y}_1$				
Layer Type	Weight Shape	Strides	Output Shape	Activation
Fully Connected	32×32	-	32×1	ReLu
Fully Connected	2×32	-	2×1	tanh
MLP[2]				
Input: $\boldsymbol{y}_0$, Output: $\boldsymbol{y}_2$				
Layer Type	Weight Shape	Strides	Output Shape	Activation
Fully Connected	32×32	-	32×1	ReLu
Fully Connected	13×32	-	13×1	Softmax
MLP[3]				
Input: $\boldsymbol{y}_0$, Output: y_3				
Layer Type	Weight Shape	Strides	Output Shape	Activation
Fully Connected	32×32	-	32×1	ReLu
Fully Connected	1×32	-	1×1	Sigmoid

F: Number of filters/kernels. K: Kernel size. Total number of parameters in CNN[1]: 5792, MLP[1]: 1122, MLP[2]: 1485, MLP[3]: 1089, Total: 9488.

was the weighted categorical cross-entropy loss. These are defined as

$$L_1 = \frac{1}{N_b'} \sum_{i=1}^{N_b} (1 - y_{3,i}) \sum_{q=1}^{2} (\boldsymbol{y}_{1,i}[q] - \hat{\boldsymbol{y}}_{1,i}[q])^2 \tag{7.78}$$

$$L_2 = -\frac{1}{N_b'} \sum_{i=1}^{N_b} (1 - y_{3,i}) \boldsymbol{y}_{2,i}^T \log(\hat{\boldsymbol{y}}_{2,i}) \tag{7.79}$$

$$L_3 = \frac{1}{N_b} \sum_{i=1}^{N_b} w_p (y_{3,i} - \hat{y}_{3,i})^2 \tag{7.80}$$

$$w_p = \begin{cases} \frac{N_{tr}}{2N_{tr}^1} & \text{if } y_{3,i} = 1 \\ \frac{N_{tr}}{2N_{tr}^0} & \text{if } y_{3,i} = 0 \end{cases} \tag{7.81}$$

$$N_b' = \sum_{i=1}^{N_b} 1 - y_{3,i} \tag{7.82}$$

where $N_b = 128$ is the batch size, i is a batch index, N_{tr}^0 is the total number of training samples with $|\delta_1| > T_{\delta_1}$ and N_{tr}^1 is the number of samples with $|\delta_1| < T_{\delta_1}$. The weight w_p gives equal average weight to the two classes in y_3. By using $1 - y_3$ as weight in L_1 and L_2, we give zero weight to samples when $|\delta_1| < T_{\delta_1}$. The total loss was the sum of L_1, L_2 and L_3. The network was trained in 40 epochs, allowing the loss to converge.

7.3.4 Tracking and Filtering Estimates

In this section we present a tracker and filtering process for the target height. This includes correcting poor estimates of n. Although we focus on the proposed CNN solution, a filtering process of this kind is necessary for both the ML and LS solutions, as these may also yield incorrect estimates of n.

We add a subscript i to denote batch index, beginning at $i = 1$ and use the notation $\hat{x}_{i|j}$ to denote an estimate of x at batch i given an estimate of x at batch j.

The filtering is split into two parts. First we filter a partial estimate of ΔN_i which we call $\Delta\nu_i$. To the filtered estimate, we then add an estimate of n_1, the value of n in the first batch. At each batch, the estimate $\hat{n}_{1|i}$ is based on all previous estimates and will thus improve over time. We then obtain an estimate of $h_{t,i}$ from the estimates of $\Delta\nu_i$ and n_1. In the second part, we filter and predict $\hat{h}_{t,i+1}$. Since we already have a filtering on $\Delta\nu_i$, this second filter would not be necessary had it not been for the possible switch to normal tracking. In such cases we instead have the phase monopulse estimate of $h_{t,i}$, and a filter based on $h_{t,i}$ is thus more convenient for switching between the two tracking modes. A more detailed description of the filter is as follows.

We split n in (7.73) in two,

$$n_i = k_i + n_1 \tag{7.83}$$

such that ΔN in (7.72) becomes

$$\begin{aligned} \Delta N_i &= \Delta n_i + n_i \\ &= \Delta n_i + k_i + n_1 \\ &= \Delta\nu_i + n_1 \end{aligned} \tag{7.84}$$

where $k_1 = 0$. From the output of MLP[1], we obtain $\Delta\hat{n}_i$ using (7.77) and compare

$$\Delta\hat{\nu}_i' = \Delta\hat{n}_i + \hat{k}_{i-1} \tag{7.85}$$

with the previous prediction, $\Delta\bar{\nu}_{i|i-1}$. If the difference between these is reduced

by increasing or decreasing $\hat{k}_{i-1}$, we update $\hat{k}_{i-1}$ accordingly, i.e.

$$l_i^\star = \underset{l\in[-1,0,1]}{\text{argmin}} \; ||\Delta\hat{\nu}_i' + l - \Delta\bar{\nu}_{i|i-1}||_2^2 \tag{7.86}$$

and

$$\hat{k}_i = \hat{k}_{i-1} + l_i^\star \tag{7.87}$$

$$\Delta\hat{\nu}_i = \Delta\hat{n}_i + \hat{k}_i \tag{7.88}$$

The filtered updates of $\Delta\nu$, $\Delta\bar{\nu}_{i|i}$ are obtained from a Kalman filter, [110], presented in Section 7.3.4.1.

To assert our assumption that $k_1 = 0$, we do not allow $l_i^\star$ to equal one, nor do we update k in (7.87) during the first batches. This is relevant when $\Delta\nu$ is close to an integer at low SNRs. In such cases the initial estimate of $\Delta\nu$ in (7.85) may be close to either zero or one, and consequently a track may begin at $k = 1$ if we allow $l_i^\star = 1$. We found that imposing this constraint for the first 10 batches was sufficient for the cases considered in this study. This number will of course depend on the initial value of the true $\Delta\nu$ and the SNR.

Once the filter has been initialized in this manner, we shift the output from MLP^2 by $-\hat{k}_i$

$$\tilde{\boldsymbol{y}}_{2,i}[j] = \begin{cases} \hat{\boldsymbol{y}}_{2,i}[j - \hat{k}_i] & \text{if } 1 \leq j - \hat{k}_i \leq n_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{7.89}$$

for $j = 1, ..., n_{\max}$, and average this with the previous shifted output of MLP^2,

$$\bar{\boldsymbol{y}}_{2,i} = \frac{i-1}{i}\bar{\boldsymbol{y}}_{2,i-1} + \frac{1}{i}\tilde{\boldsymbol{y}}_{2,i}. \tag{7.90}$$

An incremental change in $\hat{k}$ implies, at least ideally, that the argument of the maxima of $\hat{\boldsymbol{y}}_2$ is shifted by the same increment. By shifting $\hat{\boldsymbol{y}}_2$ by $-\hat{k}$ we align $\tilde{\boldsymbol{y}}_2$ with the previous estimate, $\bar{\boldsymbol{y}}_{2,i-1}$.

Let

$$\hat{n}_{1|i} = -1 + \text{argmax}\, \bar{\boldsymbol{y}}_{2,i} \tag{7.91}$$

be the estimate of n_1 at batch i. The CNN based estimate of the target height, $\hat{h}_{t,i}^{\text{CNN}}$, can then be obtained from

$$\Delta\hat{R}_i = \Delta\hat{N}_i\lambda_c = \left(\Delta\bar{\nu}_{i|i} + \hat{n}_{1|i}\right)\lambda_c \tag{7.92}$$

and $\hat{R}_{1,i}$ using the method in Section 7.3.1.

Finally we let a selector choose between $\hat{h}_{t,i}^{\text{CNN}}$ and the phase monopulse estimate of the target height

$$\hat{h}_{t,i}^{\text{MP}} = \sqrt{\hat{R}_{1,i}^2 + (\tilde{r}_e + h_r)^2 + 2\hat{R}_{1,i}(\tilde{r}_e + h_r)\sin(\bar{\theta}_{1,i}^{\text{MP}})} - \tilde{r}_e \tag{7.93}$$

where $\bar{\theta}_{1,i}^{\text{MP}}$ is the average of all estimates of θ_1 in batch i using (7.38). Our selector is

$$\hat{h}_{t,i} = \begin{cases} \hat{h}_{t,i}^{\text{CNN}} & \text{if } \bar{y}_{3,i} \leq T_{y_3} \\ \hat{h}_{t,i}^{\text{MP}} & \text{otherwise} \end{cases} \tag{7.94}$$

where $\bar{y}_3$ is the filtered $\hat{y}_3$

$$\bar{y}_{3,i} = \begin{cases} a\bar{y}_{3,i-1} + (1-a)\hat{y}_{3,i} & \text{if } \hat{y}_{3,i} < \bar{y}_{3,i-1} \\ b\bar{y}_{3,i-1} + (1-b)\hat{y}_{3,i} & \text{otherwise} \end{cases} \tag{7.95}$$

where $a \in [0,1]$, and $b \in [0,1]$. By selecting $a < b$ we lower $\bar{y}_{3,i}$ when $\hat{y}_{3,i}$ fluctuates between high and low values; indicating uncertainty, while allowing $\bar{y}_{3,i}$ to become high when $\hat{y}_{3,i}$ is consistently high.

The estimate $\hat{h}_{t,i}$ is then filtered via a second Kalman filter to produce the final predicted estimate of h_t, $\hat{h}_{t,i+1|i}$. Fig. 7.7 shows the entire estimation process of $\hat{h}_{t,i+1|i}$ from the inputs $\eta(\tilde{\boldsymbol{f}})$, $\kappa(\tilde{\boldsymbol{f}})$ and $\Sigma_x(\tilde{\boldsymbol{f}})$.

The filtering process presented here is necessary, but is not the main focus in this study. We will therefor not go into much detail on the choice of parameters in the Kalman filters nor do we make any claims on optimality with the presented filtering method.

We chose the Kalman filter since we only considered simple linear dynamics of the target. If more advanced nonlinear dynamic models were to be used, such as the one in [182], the Kalman filters could, and perhaps should, be replaced with nonlinear methods. Some examples are the extended Kalman filter, the unscented Kalman filter, the interacting multiple model algorithm [183], [184] or even neural networks such as recurrent recurrent neural networks (RNNs). This is however beyond the scope of this study.

7.3.4 The Kalman Filter and Target Dynamics

We use two Kalman filters [110], for two different variables, h_t and $\Delta\nu$. To present both filters with the same equations we define the generic state variable vector $\boldsymbol{x}^z = [x, \dot{x}]^T$ with covariance matrix $\boldsymbol{P}^z$, where z denotes the variable in question, i.e. $z = h_t$ or $z = \Delta\nu$. The state vector $\boldsymbol{x}^z$ is assumed to have the linear dynamic

model

$$\boldsymbol{x}_{i+1}^z = \boldsymbol{\Phi}_i \boldsymbol{x}_i^z + \boldsymbol{u}_i^z \tag{7.96}$$

where u_i^z is zero mean Gaussian process noise with covariance matrix

$$\boldsymbol{Q}_i^z = \begin{bmatrix} T_{b,i}^4/4 & T_{b,i}^3/2 \\ T_{b,i}^3/2 & T_{b,i}^2 \end{bmatrix} \sigma_{u,z}^2. \tag{7.97}$$

$\boldsymbol{\Phi}_i$ is the transition matrix from one state to the next and is

$$\boldsymbol{\Phi}_i = \begin{bmatrix} 1 & T_{b,i} \\ 0 & 1 \end{bmatrix} \tag{7.98}$$

where $T_{b,i}$ is the time between two consecutive batches and depends on the PRF, and consequently the range, see Table 7.1.

The filters' update steps are

$$S_i = \boldsymbol{H} \bar{\boldsymbol{P}}_{i|i-1} \boldsymbol{H}^T + \sigma_{v,z}^2 \tag{7.99}$$

$$\boldsymbol{K}_i = \bar{\boldsymbol{P}}_{i|i-1} \boldsymbol{H}^T S_i^{-1} \tag{7.100}$$

$$\bar{\boldsymbol{x}}_{i|i}^z = (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}) \bar{\boldsymbol{x}}_{i|i-1}^z + \boldsymbol{K}_i z_i^{\text{obs}} \tag{7.101}$$

$$\bar{\boldsymbol{P}}_{i|i} = (\boldsymbol{I} - \boldsymbol{K}_i \boldsymbol{H}) \bar{\boldsymbol{P}}_{i|i-1} \tag{7.102}$$

where z_i^{obs} is the observed variable at batch i

$$z_i^{\text{obs}} = \boldsymbol{H} \boldsymbol{x}_i^z + v_i^z \tag{7.103}$$

$$\boldsymbol{H} = [1, 0] \tag{7.104}$$

and v_i^z is measurement noise with variance $\sigma_{v,z}^2$. In our case this comes from the estimation error in the CNN. This is only approximately zero mean Gaussian which is the assumption for optimality in [110]. The filters' prediction steps are

$$\bar{\boldsymbol{x}}_{i+1|i}^z = \boldsymbol{\Phi}_i \bar{\boldsymbol{x}}_{i|i} \tag{7.105}$$

$$\bar{\boldsymbol{P}}_{i+1|i}^z = \boldsymbol{\Phi}_i \bar{\boldsymbol{P}}_{i,i}^z \boldsymbol{\Phi}_i^T + \boldsymbol{Q}_i^z \tag{7.106}$$

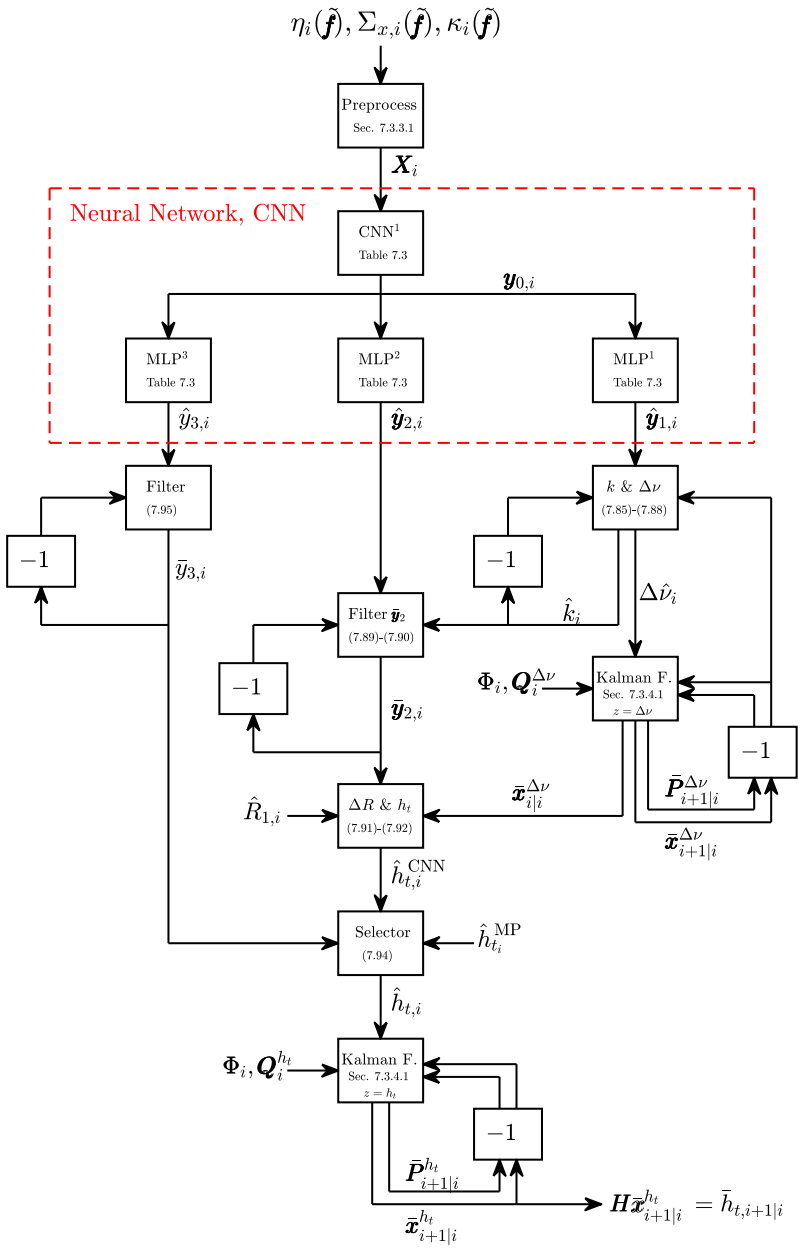


Figure 7.7: Block diagram of the CNN and tracking filter on a batch level. The blocks marked -1 signify a delay of one.

7.4 Results

The presented results are on the test set as well as some specific cases. The test set was randomly generated using the parameters in Table 7.2 except for the SNR. In all cases the radar height was 5 m.

7.4.1 Comparison with ML and LS methods

Different search methods for finding the path difference between grazing and direct paths are discussed in [185]. A search method based on alternating projections for the ML optimization in low angle estimation is discussed in [186]. This is a computationally more effective method than a simple grid search in each dimension, but does not always work, [187]. For the sake of simplicity we use a grid search. Although computationally heavy, this is a robust alternative provided a fine enough grid.

The ML and LS solutions were obtained from a 500 point search in h_t within the range $[0, 500]$ m. At the best point, an additional search of 100 points was performed between the neighboring points of the maximum. This yields a quantisation resolution well below the obtained variance in h_t. The ML solution also had two grid searches in $|\Gamma|$ and $\tilde{\sigma}$, with 10 points for Γ uniformly distributed between 0 and 1 and 30 points for $\tilde{\sigma}$ between -50 and 40 dB. Increasing the number of grid points did not have a noticeable effect while reducing them by half, gave significant increase in the RMSE. The LS solution on $\eta(\tilde{\boldsymbol{f}})$ had the same search in $|\Gamma|$. Performing the search with the true value of the argument of Γ rather than assuming that $\angle\Gamma = \pi$ gave negligible improvement.

The computational time for each solution when evaluated in MATLAB is shown in Table 7.4. The term batch in this case refers to one batch of N_f samples producing one estimate of h_t - not to be confused with the batch in the training process. These values depend on many factors such as the machine hardware, but the results in Table 7.4 still give an indication of the relative computational cost between the different methods. Although the numbers for the ML and LS solutions can be reduced by a more effective optimisation technique, we note that this reduction would need to be considerable to reach the computational time of the CNN, at least for the ML method. Regarding the CNN, we have not exploited parallelisation. Evaluating all 1000 test samples in parallel took 0.02 seconds with an additional 0.03 seconds for the preprocessing.

Fig. 7.8 shows the RMSE of the estimate of all h_t for the different methods. We see, perhaps somewhat surprisingly, that the CNN performs equally or better than the ML solution up to 55 dB SNR. This is mainly due to the assumption that Γ is constant over frequency. The relation between Γ and h_t has also not been

Table 7.4: Batch Execution Time

Method	**Execution Time**	**Execution Time Per Iteration**
ML on $\eta(\tilde{\boldsymbol{f}})$	10 s	70 μs
LS on $\eta(\tilde{\boldsymbol{f}})$	0.36 s	70 μs
LS on $\lvert\Sigma_x(\tilde{\boldsymbol{f}})\rvert$	4.5 ms	7.5 μs
CNN	0.4 ms	-

Evaluated with a six core Intel® Core™ i7-8750H CPU at 2.2 GHz.

taken into account. If we would have used the model for the reflection coefficient in Section 7.2.2 in the grid search, and thus optimize over the significant wave height instead of $|\Gamma|$, the RMSE would be approximately at the level of the CNN. This solution depends heavily on the used model and a slight mismatch between the model and a possibly different reality may cause catastrophic failure in the optimisation. Since the CNN appears to exploit the relation between $|\Gamma|$ and frequency (as well as h_t), one may wonder if the CNN will also fail when evaluated on different data or models. The answer to this is yes, as we will see later in this section. There difference is however that a ML and LS solutions are dependent on a mathematical model, whereas the CNN solution is dependent on data.

An additional source of error is that the model for the ML and LS assumes that all parameters are constant over all pulses in the batch. This is however not the case in the data generation model, where the parameters change somewhat due to the target velocity. The variance of the error between this assumption, and the case when it is true, corresponds to adding noise with approximately 40 dB SNR. This is an average and depends on range, height and velocity, and will not necessarily be complex Gaussian. Due to this error, we do not see much improvement at higher SNRs.

We note in Fig. 7.8 a significant improvement when $\hat{n} = n$, i.e. when the correct lobe is chosen. In this case the RMSE of the CNN converges to 1.45 m. The improvement is apparent for all four methods showing that all suffer from poor estimates of n.

In Fig. 7.8 we see two curves for the CNN. One shows the CNN as described in Section 7.3.3, the other shows the case where only $\eta(\tilde{f})$ is used as input. We note that most of the information lies in $\eta(\tilde{f})$, which is also apparent when comparing the two LS methods. Adding $|\Sigma(\tilde{f})|^2$ to the CNN input does however yield a slight improvement.

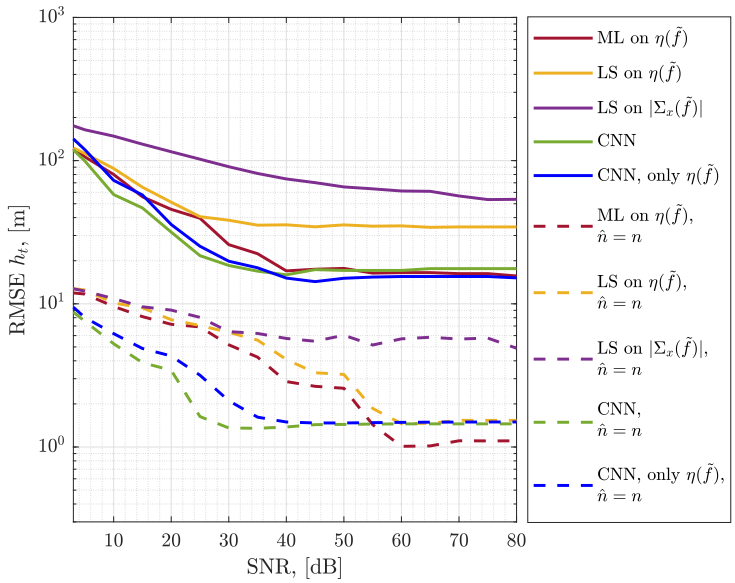


Figure 7.8: RMSE of h_t as a function of the SNR. Due to assumption errors, the actual SNR is limited to approximately 40 dB, hence the lack of improvement at higher SNRs. The dashed lines show the RMSE if $\hat{n} = n$, i.e. when the correct lobe is selected.

7.4.2 Filtering the Estimate of n

Fig. 7.9-7.11 illustrates the principle of filtering out incorrect n. In Fig.7.9 we see the estimate of $\Delta\nu = \Delta n + k$ for different values of $\hat{k}$. The filtered estimate of $\Delta\nu$ follows a natural trend and $\hat{k}$ is updated at each integer crossing rather than decreasing $\Delta\hat{\nu}$ by an integer. What remains is to estimate n_1. This is of course the same during the whole track, and by aligning the batch based estimates of $\boldsymbol{y}_{2,i}$ with $\boldsymbol{y}_{2,1}$ using $\hat{k}_i$, we can base the current estimate of n_1 on all previous batches, see Fig. 7.10. By doing so we obtain a close to 100% accuracy on n_1 and consequently on n. In the case in Fig. 7.9-7.11 the estimate of n_1 was only incorrect for the first 11 out of the 2425 batches. By estimating n_1 separately and not including it in $\Delta\nu$, we avoid locking on to incorrect n in the filtering process. Recovering from an initially incorrect estimates of n_1 is illustrated in Fig. 7.11.

The low accuracy on $\hat{\boldsymbol{y}}_2$ in Fig. 7.10 at 6.2 km and 4.3 km is due to poor SNR since at these ranges the interference is approximately at a null over the whole bandwidth.

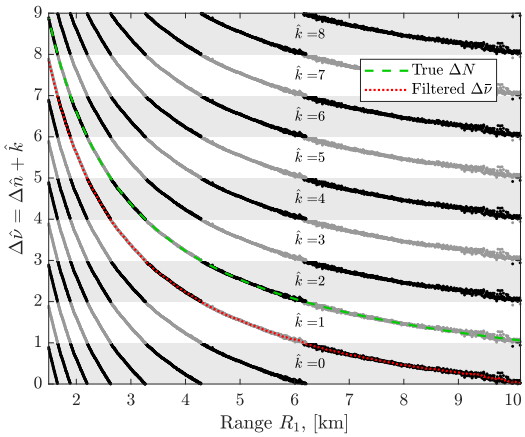


Figure 7.9: Estimate of $\Delta\nu$ for different values of k over a track. The red dashed line shows the track starting at $k = 0$. This differs by one from the true track of ΔN, marked with a green dashed line. The initial SNR in this case was 3 dB and increased 40 dB per decade in range.

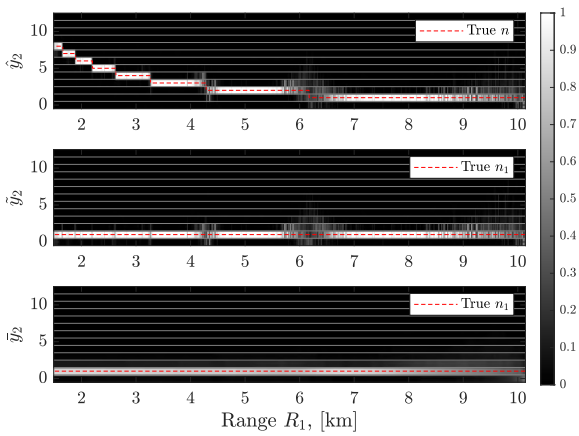


Figure 7.10: Estimate of the initial n from $\hat{\boldsymbol{y}}_2$ and $\hat{k}$. By shifting $\hat{\boldsymbol{y}}_2$ (top plot) by the estimated k, the argument of the maximum in $\tilde{\boldsymbol{y}}_2$ (middle plot) aligns on average at the value of n_1. By averaging with previous estimates, the argument of the maximum of $\bar{\boldsymbol{y}}_2$ (bottom plot) is at the true value of n_1 almost throughout the entire track, and consequently the correct value of $n = k + n_1$, dashed line in top plot, is obtained. The initial SNR in this case was 3 dB and increased 40 dB per decade in range.

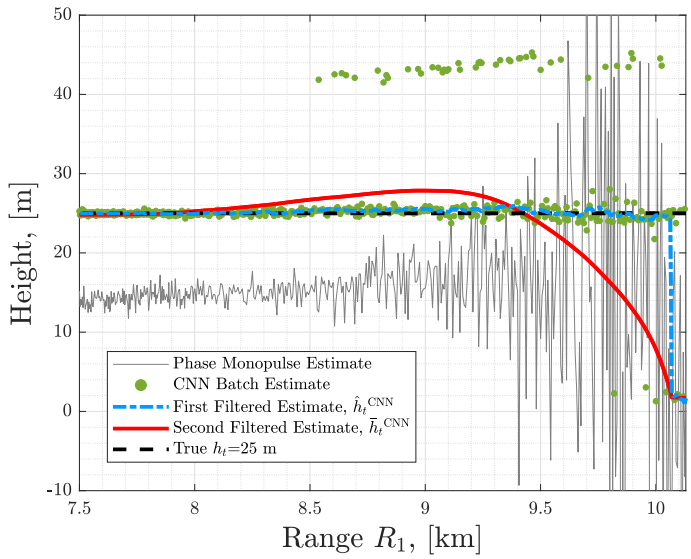


Figure 7.11: Example of recovery from incorrect estimates of n. The true value of n_1 is one. In this case the estimate of n_1 was zero for the first 11 batches, resulting in a far too low estimate of h_t. Once the score for $\hat{n}_1 = 1$ becomes dominant, the first estimate of h_t, $\hat{h}_t^{\text{CNN}}$ immediately switches since the filtering is only on $\Delta\nu$ and does not involve n_1. The second Kalman filter output, $\bar{h}_i^{\text{CNN}}$, is on $\hat{h}_i^{\text{CNN}}$ and takes longer time to adapt to the change in $\hat{n}_1$.

7.4.3 Normal Tracking and Multipath Tracking

Here we take a closer look at the tracking of h_t. For this we use the parameter values in Table 7.5 and the two scenarios in Table 7.6. The variances of the measurement noise, $\sigma^2_{u,\Delta\nu}$ and σ^2_{u,h_t} are the corresponding MSE values in the training and validation data (these did not differ by much).

Fig. 7.12 shows the effect of varying the threshold T_{y_3} for different SNR values. We note that the threshold should depend on the SNR. We have however selected $T_{y_3} = 0.6$ in the following results.

Fig. 7.13 shows the tracking of h_t for the two cases. We clearly see the oscillatory effect of multipath interference in the phase monopulse estimate. In case 2 the combination of target height and sea state is such that $|\delta_1|$ becomes very small at ranges less than 4 km. At this point the CNN fails but the tracker switches to normal tracking and the height estimate is unaffected.

Table 7.5: Filter Parameters

Parameter	**Value**
a	0.9
b	0.99
T_{y_3}	0.6
$\sigma^2_{u,\Delta\nu}$	1000
σ^2_{u,h_t}	1000
$\sigma^2_{v,\Delta\nu}$	0.02
σ^2_{v,h_t}	20

Table 7.6: Scenario Parameters

Case 1	**Case 2**
$v_t = 1000$ m/s	$v_t = 1000$ m/s
Initial SNR = 10 dB	Initial SNR = 10 dB
$H_{1/3} = 0.1$ m	$H_{1/3} = 1$ m
$h_t = 5$ m	$h_t = 50$ m

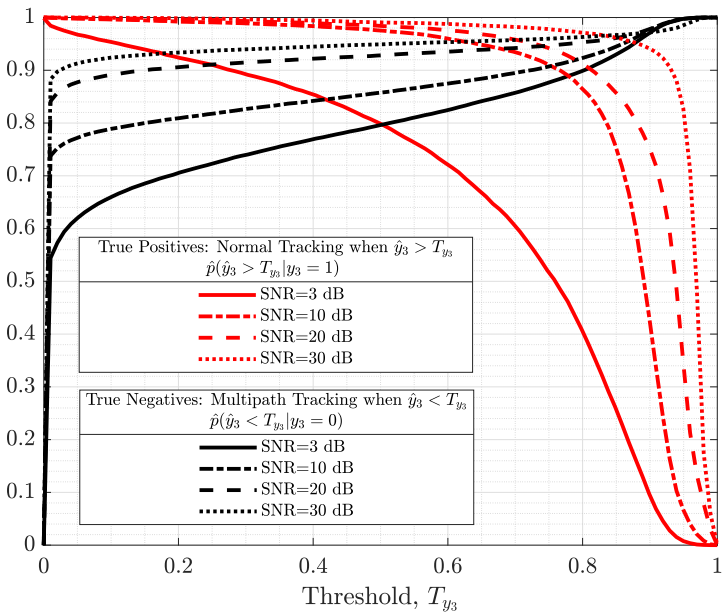


Figure 7.12: Empirical receiver operating characteristic for the estimate of the normal tracking indicator y_3 using the test data.

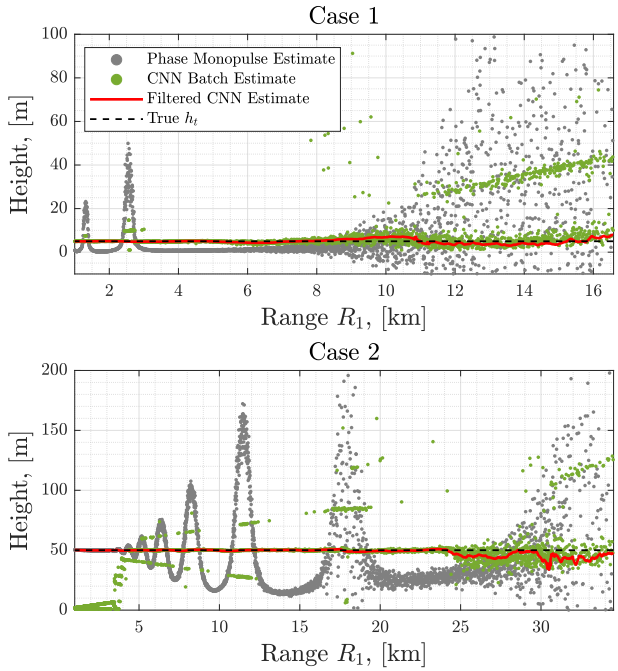


Figure 7.13: Filtered CNN estimates for the two benchmark scenarios. For case 2 the tracker switches to normal tracking at 3.8 km range where the CNN estimate begins to go towards $h_t = 0$ due to small $|\delta_1|$.

7.4.4 Comparison with Direction of Arrival Methods

As discussed in Section 7.1, an alternative to the presented interferometric approach is the angle estimating direction of arrival (DOA) approach. We compare these with regards to simulation results as well as the Cramér Rao bound (CRB) for each method. The CRB gives a lower bound for the MSE of an unbiased estimator and is defined as

$$\mathrm{CRB}(\boldsymbol{\beta}) = \boldsymbol{I}(\boldsymbol{\beta})^{-1} \tag{7.107}$$

where $\boldsymbol{I}(\boldsymbol{\beta})$ is the Fisher information matrix with elements

$$\boldsymbol{I}(\boldsymbol{\beta})[i,j] = \mathbb{E}_{\boldsymbol{x}}\left[\frac{\partial^2 \log p(\boldsymbol{x}|\boldsymbol{\beta})}{\partial \boldsymbol{\beta}_i \partial \boldsymbol{\beta}_j}\right] \tag{7.108}$$

where $p(\boldsymbol{x}|\boldsymbol{\beta})$ is the likelihood function. Using $\eta(\tilde{\boldsymbol{f}})$, we obtain the CRB for the parameter h_t from (7.107) and (7.108) with the likelihood in (7.47). Here we calculate the expected value numerically and we have assumed that

$$\frac{\partial \theta_1}{\partial h_t} \approx \frac{1}{R_1} \tag{7.109}$$

$$\frac{\partial \theta_2}{\partial h_t} \approx -\frac{1}{R_1} \tag{7.110}$$

$$\frac{\partial \Delta R}{\partial h_t} \approx \frac{2h_r}{R_1}. \tag{7.111}$$

As for the CRB for DOA methods, we first note that with a digital antenna array, we can rewrite (7.24) as

$$\boldsymbol{s}(\tilde{f}) = \boldsymbol{A}\boldsymbol{z} + \boldsymbol{u} \tag{7.112}$$

where

$$\boldsymbol{A} = [\boldsymbol{a}(\theta_1), \boldsymbol{a}(\theta_2)] \tag{7.113}$$

$$\boldsymbol{a}(\theta)[i] = e^{j2\pi d_i \sin(\theta)\tilde{f}/\lambda_c} \tag{7.114}$$

$$\boldsymbol{z} = 2|C|\frac{\sqrt{\sigma}}{R_1^2} e^{j(\phi+\angle C)} G_I(\theta_1) F(\delta_1) \begin{bmatrix} 1 \\ \Gamma e^{-j\varphi} \end{bmatrix} \tag{7.115}$$

$$\varphi = 2\pi \Delta R \tilde{f}/\lambda_c \tag{7.116}$$

$$\boldsymbol{u} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I}) \tag{7.117}$$

when every element of $\boldsymbol{s} \in \mathbb{C}^{2M\times 1}$ is sampled at $t \approx \tau_1$. The angle $\phi + \angle C$ may be regarded as stochastic. Our sampled signal $\boldsymbol{s}(\tilde{f})$ will then have covariance matrix

$$\boldsymbol{R_s} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^* + \sigma^2 \boldsymbol{I} \tag{7.118}$$

where * signifies transpose conjugate and

$$\boldsymbol{P} = 4\frac{|C|^2 G_I(\theta_1)^2 |F(\delta_1)|^2 \sigma}{R_1^4} \begin{bmatrix} 1 & \Gamma^* e^{j\varphi} \\ \Gamma e^{-j\varphi} & |\Gamma|^2 \end{bmatrix}. \tag{7.119}$$

Note that $\boldsymbol{P}$ is a rank one matrix due to the coherent paths. The stochastic CRB for (7.119) is derived in [188]. For simplicity we let $\tilde{f} = 1$ for all N_f pulses. The CRB for case 1 in Table 7.6 is shown in Fig. 7.14 for both the interferometric and DOA method. We show the MSE of 1000 runs using ESPRIT with and without

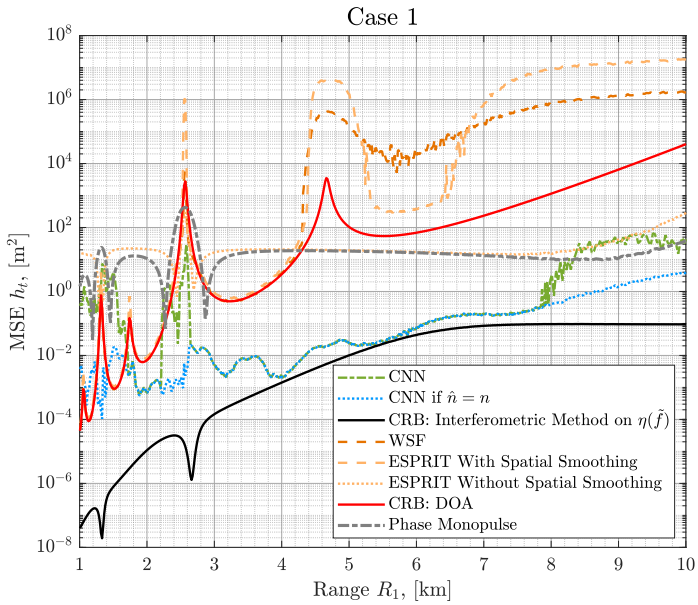


Figure 7.14: MSE of h_t for case 1 in Table 7.6 from 1000 Monte Carlo runs.

spatial smoothing, WSF and the CNN solution. For the spatial smoothing we used ten subarrays, resulting in a covariance matrix with size 55×55. For reference, we also show the phase monopulse MSE. The phase monopulse, as well as ESPRIT without spatial smoothing, are not unbiased estimators of θ_1, and the DOA CRB does therefore not apply to these.

We note that the interferometric CNN solution lies well below the CRB for DOA methods for ranges beyond 2 km. We also clearly see the effect of spatial smoothing for ESPRIT. The WSF method attains the CRB at about 4.2 km range. This range can be moved further out with better SNR, larger batch size or higher h_t (i.e. larger angle separation). However reducing the MSE of the DOA methods to the level of the CNN would require a gain of about five orders of magnitude. Also note that for the DOA methods we have assumed a digital array with 64 channels and not the two channel system described in Section 7.2.3.

7.4.5 Alternative Models

The performance of the CNN has so far only been evaluated on data that is within the distribution of the training data. We now look at some examples where this is

not the case.

7.4.5 Clutter and Noise Models

In our model we have assumed Gaussian receiver noise. A more realistic model would also include sea clutter reflections. The compound K-distribution, presented in [189], has been empirically validated as a good model for sea clutter data [190]. A shape parameter ν_{K}, determines the length of the tails of the clutter distribution. Typical values are $0.1 < \nu_{\mathrm{K}} < 10$, [191], where small ν_{K} implies long tails or very "spikey" clutter.

Fig. 7.15 shows the RMSE of h_t in the test set with Gaussian noise (same as the results in Fig. 7.8), and with K-distributed noise with shape parameter $\nu_{\mathrm{K}} = 10$ and two cases with $\nu_{\mathrm{K}} = 0$. These are with uncorrelated clutter and with rather high correlation over frequency. This correlation was generated according to the method described in [192]. As seen, there is very little difference between these for a given SNR or signal-to-clutter ratio (SCR). As for the correlated clutter, the patterns within the bandwidth are still much more random and of higher frequency content than the very low frequency patterns resulting from multipath. See Fig. 7.5. Correlation between sea clutter returns in frequency agile radar will also depend on parameters such as pulse lengths and is further discussed in [7, Sec. 7.6].

7.4.5 The value of N_f

In our evaluations we have used 50 pulses in each batch. Fig 7.16 shows the RMSE of h_t as a function of N_f, again assuming that $\hat{n} = n$. For higher SNRs, N_f could be reduced to 16, i.e. one pulse per frequency band, without much loss. At lower SNRs the gain of the non-coherent pulse integration is more apparent. Larger batch sizes also entail longer paths covered by the target during measurement. This introduces more error on the assumption that ΔR is constant for all pulses in the batch.

In the training data we have assumed target velocities in the range 100 - 1000 m/s. Hypersonic missiles will however fly faster than this, e.g. [193] reports future cruising speeds greater than Mach 5, or 1700 m/s. Since the range covered by the target during a batch is

$$\varepsilon_{R_1} = \frac{v_r(N_f - 1)}{\mathrm{PRF}}, \tag{7.120}$$

a change in v_r can be compensated for by either changing the PRF, or N_f. As we saw in Fig 7.16, changing N_f will have little effect, at least for higher SNRs.

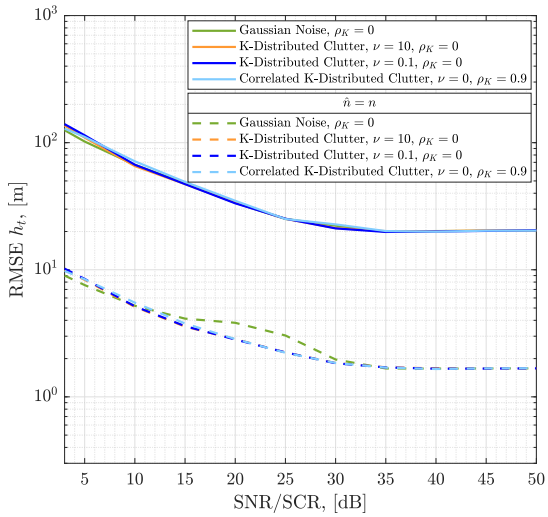


Figure 7.15: RMSE of h_t in the test set for different noise/clutter models. The bottom curves are for the case when the correct lobe is estimated, i.e. $\hat{n}$ equals the true value. ρ_K denotes the correlation coefficient when shifting frequency 32 MHz (total bandwidth divided by N_f).

7.4.5 Reflection Coefficient Models

Next we consider the model for the reflection coefficient Γ in (7.12). A different model for the surface roughness factor is presented in [194] which is more tuned to empirical data. In addition, this model also includes a diffuse component in the reflection coefficient. This accounts for reflections from multiple waves and results in a more randomized phase in higher sea states. Since our detection method is based on phase difference over frequency, we can expect poorer performance using this model. Fig. 7.17 shows the RMSE of h_t assuming $\hat{n} = n$ and the accuracy on n for the two different models. At lower sea states, $H_{1/3} < 1$ or Douglas sea state 0-3, there is little difference, at least for h_t. At higher sea states on the other hand, the increase in error is significant. At the same time this randomisation will have an averaging effect on the multipath interference and therefore presents less problems for normal tracking. One may also question the feasibility of sea skimming in rough sea states.

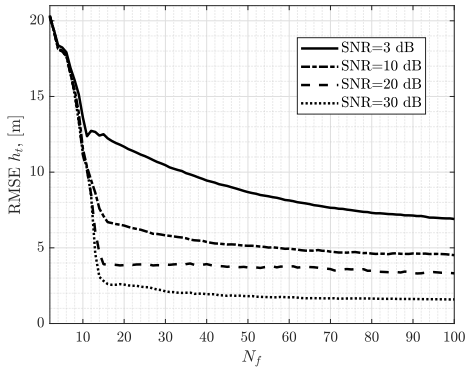


Figure 7.16: RMSE of h_t as a function of the number of pulses per batch N_f when $\hat{n} = n$. For reference, the RMSE for guessing $\Delta n = 0.5$ when $\hat{n} = n$, is 20 m for this dataset.

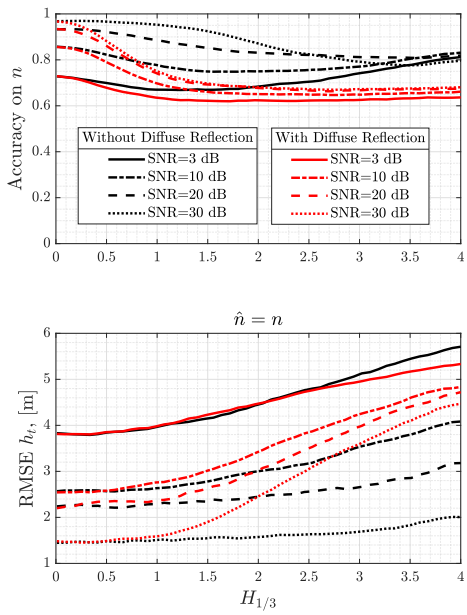


Figure 7.17: Accuracy on n and RMSE of h_t when $\hat{n} = n$ using the test set parameters but with different reflection models. The model presented in Section 7.2.2 is without diffuse reflection.

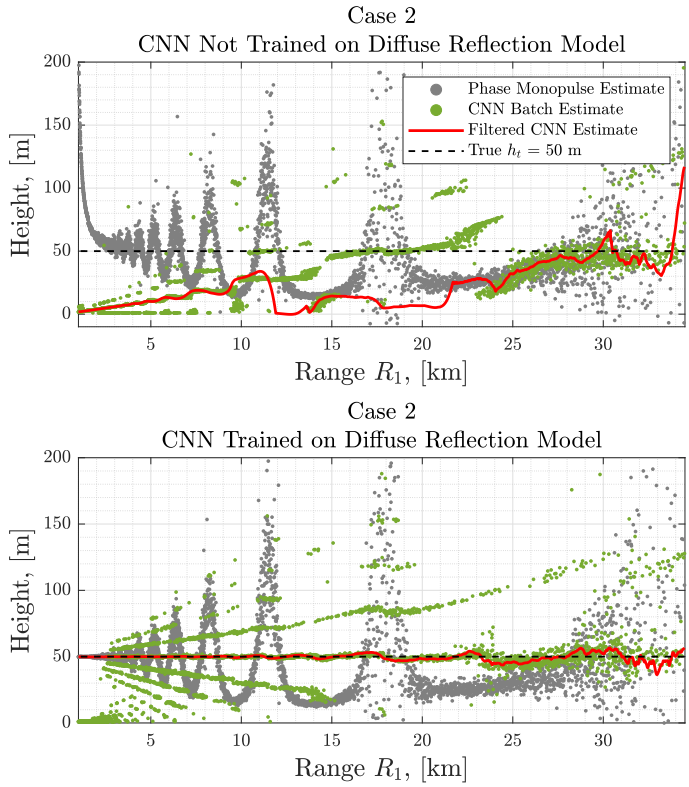


Figure 7.18: Case 2 in Table 7.6 with a different reflection model. The top plot is with the CNN trained on the reflection model presented in Section 7.2.2. This diverges slightly at 25 km causing a tracking failure. In the bottom plot the CNN has been trained on the different reflection model and does not lose the track. The increased uncertainty in n is clear when comparing the batch estimate with case 2 in Fig. 7.13.

The error introduced by this different reflection model may however be large enough to lose track of the target. An example of this is shown in Fig. 7.18. On the other hand, if trained on the diffuse reflection model, the CNN performs well again. For the lower sea state in case 1, the tracking performance was unaffected when the diffuse reflection model was used.

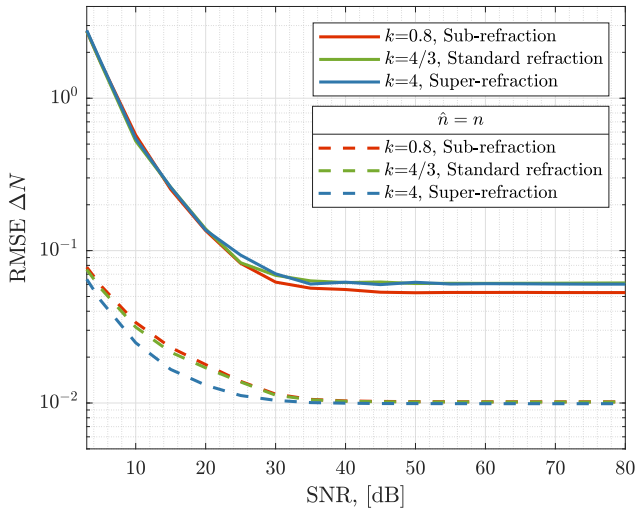


Figure 7.19: RMSE of the CNN estimate of the normalized path difference for different values of the refraction factor k. The bottom curves are for the case when the correct lobe is estimated, i.e. $\hat{n}$ equals the true value.

7.4.5 The Effect of the Refraction Factor k

As stated in Section 7.2.1, we have assumed a refraction factor of k =4/3 throughout this study. This is sometimes referred to as standard refraction [178]. Depending on weather conditions however, the actual refraction factor may deviate significantly from this value, being both smaller (sub-refraction) or larger (super-refraction). In [178, Sec. 5.2-5.3], values of $k = 0.8$ or less, and up to and beyond $k = 4$ are reported. This will have a significant effect on the estimate of h_t, since the geometric relation used for producing this estimate is determined by k.

It should be pointed out however, that the mapping from the input data $\boldsymbol{X}$ to the path difference ΔR will be practically the same regardless of the value of k. A dependence on k comes from the dependence of k in the grazing angle, but the effect of this is small. Since the CNN is only used to estimate ΔR, we can use the same network for any k, and instead tune the mapping from ΔR to h_t depending on k. This is an advantage of not having an end to end solution where we would estimate h_t with a CNN directly. Fig. 7.19 shows the RMSE of the estimated normalized path difference $\Delta N = \Delta R/\lambda_c$ for different values of k. We see that the effect of varying k is indeed small, and for high SNRs there is hardly any effect at all when the correct lobe is estimated.

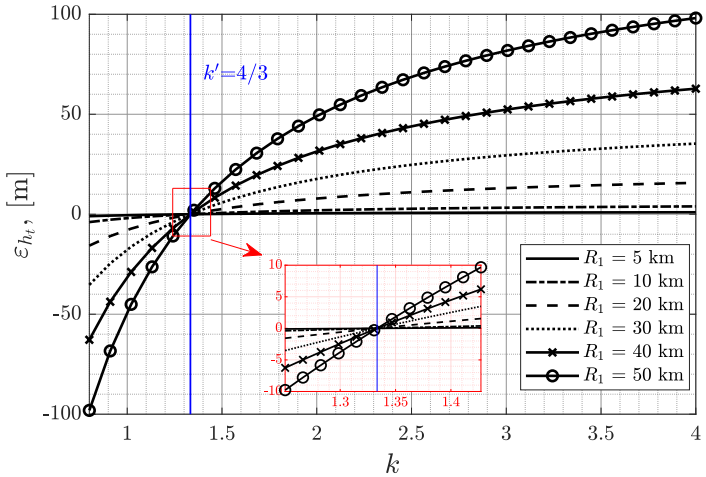


Figure 7.20: The upper bound in magnitude of ε_{h_t} as a function of the refraction factor k when the value k =4/3 is assumed. This bound is more exaggerated for low values of h_t. As an example, assume that the true value of k is 2.5. This gives $\varepsilon_{h_t} = 11$ m at 20 km range. If we do not utilize the approximation in (7.11), the calculated value of ε_{h_t} is for this case 10 m if $h_t = 100$ m, but 6 m if $h_t = 5$ m.

These results assume that the radar, target and sea surface are all within the same duct. The height of such ducts can vary between a few meters and couple of hundred meters [176]. When the target and/or radar are above the surface duct, the apparent reflection coefficient may differ from the model presented in Section 7.2.2. As we saw earlier here in Section 7.4.5.3, a mismatch between the data that the CNN is trained on, and the data it is evaluated on, may yield poor results, as is often the case with CNNs.

Next we look at the height error when using an incorrect value of k. Assume that we have a system that can perfectly estimate the angle of arrival θ_1. Using the approximation in (7.11) we can calculate the target height as

$$h_t' \approx R_1 \sin(\theta_1) + \frac{R_1^2}{2k' r_e} + h_r \tag{7.121}$$

where we have written out $\tilde{r}_e = kr_e$, and k' is the assumed refraction factor. Assume now that k' differs from the true refraction factor k. The difference

between h_t' in (7.121) and the calculated height using the true k is

$$\varepsilon_{h_t} \approx R_1^2 \frac{k - k'}{2kk'r_e} \tag{7.122}$$

This approximation is also an upper bound to the introduced bias/error. Fig. 7.20 shows some values of ε_{h_t} as a function of k for different ranges when $k' = 4/3$. At larger ranges we see that even slight variations in k introduces a bias in h_t that quickly becomes larger than the RMSE of h_t shown in Fig. 7.8. An accurate estimate of k is therefore essential for this type of geometric based low angle tracking. At least for long ranges. We also note that k is an important parameter to estimate for the operation of naval radar systems in general, and not only for the tracking scheme presented in this study. Estimating k is not trivial. A method for this is presented in [178, Sec. 5.3].

7.5 Conclusions

We have in this study shown that a small CNN can be used for elevation tracking of sea skimming targets with a performance comparable to or better than the presented ML and LS methods. An advantage of such a solution is the relatively low computational cost of estimating the target height.

Although we have used a specific model in our simulations, the presented CNN solution can just as well be applied to real data and/or other models with different clutter and receiver statistics, without changing the method. As is the case with ML methods, model errors may cause significant errors [195]. We have shown that the CNN solution could be sensitive to changes in the data distribution and it is therefore important to train the CNN on data with high fidelity to real data.

The CNN as well as the ML and LS solutions suffer from estimates of the wrong multipath "beam", n. For this we have presented a simple but effective Kalman based filter for filtering out incorrect n. By so doing, the final RMSE of the height estimate was significantly reduced. The presented filtering method could however be replaced with RNNs. This could possibly yield better tracking performance in more difficult reflection models and more nonlinear target trajectories. Further analysis of this is left for future work.

Incorporated into the CNN, we have also presented a method for switching between normal tracking and multipath based tracking.

We have compared the interferometric approach of the CNN with DOA methods and shown that the interferometric approach can significantly outperform the DOA methods. DOA methods are however not without merits and a possible merge between these two methods could be a promising solution. Further analysis on

this as well as a possible merge of the ML methods with the CNN method is left for future work. The CNN estimate could for instance be used as an initial guess.

We note in Fig. 7.13 a slight bias in the height estimate, at least at long ranges. As with most data driven methods, the CNN is determined by the distribution of the training data. The shown bias can therefore possibly be adjusted by sampling the model differently. Optimizing the distribution of the training data and evaluations on actual measurements is left for future work.

We have not discussed scenarios where the target changes height during the track. Such an analysis would require assumptions on target dynamics and was left out of the scope of this study. We therefore leave this for future work. As long as the parameter ΔR can be considered approximately constant during a batch, motion in height will not significantly affect the CNN estimates. The presented tracker may however need adjustments.

Acknowledgment

We thank Anders Nordsjö and Robert Jonsson for valuable insights and discussions during this study.

Chapter 8

Conclusions and Future Work

Conclusions specific to the studies presented in Chapters 4-7 are given in these chapters, respectively. The conclusions stated in this chapter concerns the thesis as a whole and are on a more general level.

8.1 Conclusions

ANNs are versatile functions that are of great use in radar signal processing, and signal processing in general. This has been shown in multiple studies and only a fraction have been mentioned in this thesis. In our studies, we have used ANNs as classifiers and regression models for high dimensional nonlinear functions. The problems described in the four studies in Chapters 4-7 relate to different steps in a typical radar signal processing chain. Although these are quite different, a similar method has been used in all studies, namely training an ANN to map some input data $\boldsymbol{x}$ to some output $\boldsymbol{y}$ using the relatively standard optimisation methods described in Section 3.3. As is often the case with optimisation problems in general, the main focus has instead been to formulate the problem such that a mapping to $\boldsymbol{y}$ from $\boldsymbol{x}$ is feasible.

We noted in Chapter 4 that solutions to phase retrieval problems exists under certain conditions and therefore incorporated these into the formulations of $\boldsymbol{x}$ and $\boldsymbol{y}$. In Chapter 5 we noted that near perfect detection and false alarm trade-off is achievable if the integration over samples is done properly, in this case along the true target trajectory with sufficient length. These trajectories or patterns are often observable to the human eye, and we therefore assumed that is should be possible to trace these out automatically. In Chapter 6, we noted that moving parts of birds and humans will have different velocities than the moving parts of

drones. The spectrum of these velocities, obtained with micro-Doppler radar, can therefore be used as discriminating features between these different classes. Finally in Chapter 7, we noted that by estimating the path difference ΔR in a particular manner, and combining this with temporal filtering, we could remove most of the ambiguities/errors in the target height estimate that arise in single batch estimates.

Another important factor we have considered is overfitting, or how to formulate the problem such that the ANN trained on sampled data $\boldsymbol{x}_1, ..., \boldsymbol{x}_N$, and $\boldsymbol{y}_1, ..., \boldsymbol{y}_N$, will represent the desired function for general samples $\mathbf{x}$ and $\mathbf{y}$. For this we have exploited models. One of the most effective methods of ensuring that the function obtained with an ANN accurately represents the desired function, is to train the ANN on a sufficiently large set of training data. In these cases, the difference between the loss obtained from the training data, and the loss obtained from the test data will be small. This is always possible to achieve when training on data generated from models since we can collect arbitrarily large datasets with the same amount of effort. We have thereby shifted the problem from data collection, to data modeling. For this we have exploited much existing theory in radar signal processing. In this sense our approaches have been closer to traditional signal processing schemes where models are first formulated, from which analytical solutions are then derived. Here we instead use the models to shape ANNs.

As mentioned in Chapter 1, this approach of using ANNs with models has several benefits compared to more analytical methods. This includes cases where the models may

- **be too complicated to form analytical solutions**. Apart from the trivial LS solution, this has been the case in all four studies presented in Chapters 4-7, i.e. finding an inverse from $\boldsymbol{x}$ to $\boldsymbol{y}$.

- **be too simplified**. Although the used models have been simplified to some extent, this has mostly been for the purpose of executing them efficiently and not for aiding any analytical "digestion."

- **be too incorrect such that practical results differ significantly from what was theoretically expected**. The model used for drone propellers in Chapter 6 differed from what we observed in real data. We showed that basing a classifier on such a model could yield poor results even when using an ANN, see Fig. 6.6. In spite of this, we could still use the model data to improve the results by combining it with real data in the training process, and by using a model trained autoencoder as a general feature extractor. The ability to combine real data and model data is a great advantage with ANN based methods.

- **be unknown in the sense that one of several known models or parameter**

values may fit the data, but we do not know which. As an example, reducing the number of scatterers to two in the model used in Chapter 4, would results in the frequency estimation problem we saw in Chapter 7. This would however have a much higher frequency than what we saw in Chapter 7, making conventional frequency estimation methods applicable such as the DFT. Since the number of scatterers is unknown, we can however not resort to such a solution. The proposed CNN solution does not rely on such specific assumptions or constraints. We have in all four studies trained the ANNs on a relatively broad range of parameter values in order to capture all types of expected data.

- **be too complex, such that the solution will be too heavy to compute**. We examined this in Chapter 7 by comparing the CNN solution with computationally heavy LS and ML methods, as well as in Chapter 5 by comparing the CNN solution with an ideal but practically unfeasible "filter bank" of true trajectories.

The ANNs used in this thesis are small compared to many of the networks used in more general computer vision tasks. An example is the AlexNet with over 62 million parameters [73]. The reason for smaller networks is twofold. First, a motivating factor for using ANNs is to reduce the computation time. The presented signal processing tasks are typically associated with computational constraints for real time processing. By reducing the networks sizes we reduce the processing costs.

We found in all studies that significantly increasing the number of layers give either marginal improvements in the loss, or would even give poorer results in terms of training/test loss divergence. The reason for this is overfitting to noise. We found that a good number of layers was between 5 and 10 for the range of SNRs used in this thesis (-3 to 30 dB depending on study). With deeper networks we began to see significant overfitting to the noise. One could also address this by generating more training data. We found however that the gain was then often quite limited, if even noticeable. All the presented networks architectures in Chapters 4-7 are the result of much testing. The SNR sensitivity of deeper networks is the second reason for choosing smaller networks.

We discussed in Chapter 4 the trade-offs of training on different SNR ranges. These observations are also applicable to our other studies, i.e. training an ANN on high SNR values only, did not significantly improve the results on high SNRs when compared to an ANN trained on a wide range of SNRs. A network trained only on high SNRs would however perform significantly poorer on low SNR samples than a network trained with the wider SNR range. The same can be said about only training on low SNRs. An exception however is the autoencoder network

in Section 6.4. This produced more distinct projections in the latent space when limiting the SNR range in the training data to higher values.

To summarise, we have in this thesis demonstrated a new method for achieving non-coherent pulse compression. We have demonstrated a new TBD method for detection of weak signals. We have shown that CNN based classifiers on drones can be sensitive to the diversity of the training data. We have shown how model data can be used to reduce this sensitivity. We have also presented a new anomaly based classification method that relies on model data in training, yet performs well on real data. Finally, we have presented a new ANN based method for mitigating multipath interference when tracking sea skimming targets. We have shown that the performance of this is comparable to the in some sense optimal LS and ML methods, but with reduced complexity and increased flexibility to the used data/model.

8.2 Future work

The results presented in this thesis are limited by several factors. The one we deem the most critical is the fidelity of the models to real data. This has been discussed to a limited extent in Chapter 5 and Chapter 6 where we have trained on model data and evaluated on real data. In general however, the methods presented in this theses need more evaluations on real sensor data. We have focused on processing on a batch level. Extending these methods to more complex time varying scenarios is also left for future work.

ANNs are in a certain sense black boxes, in that it is difficult to analyse the functional form. We have therefore mostly relied on limited empirical evaluations and the results are limited to these. It is however questionable whether more such evaluations would be of much value until we have verified whether or not the models, on which we have based the evaluations, are representative of real sensor data.

8.3 Reflections

There is on the island of Visingsö in the lake of Vättern in Sweden, a large grove of oak trees. Commissioned by the Swedish navy, these trees were planted in 1831 to be used as timber for building ships. Once ready, some 150 years later, shipbuilding techniques had long since advanced and the oak trees had become obsolete. Still standing today, they may be regarded as a reminder of the importance of preparing for the future, or more specifically, preparing for the risk of future conflicts.

Some readers may have ethical concerns about the research in this thesis. The

research has in part been funded by Saab, and it is thus likely that the presented methods and results may be used to develop weapons in some form. We have however seen during the course of the presented studies, 2018-2023, the rise of pandemics, conflicts and wars that to many were not easy to foresee, even as they were unfolding. We cannot know what kind of situations the future has in hold, or what the armament needs will be. We can only conclude that the answer to this may not be our choice, and that the "timber" of the future needs to be planted today. To this end we present the acorn that is this thesis.

References

[1] "IEEE standard for radar definitions," *IEEE Std 686-2017 (Revision of IEEE Std 686-2008)*, pp. 1–54, 2017. doi: 10.1109/IEEESTD.2017.8048479

[2] T. Johnsen and K. E. Olsen, "Bi- and multistatic radar," *The NATO Science and Technology Organization*, 2007. doi: 10.14339/RTO-EN-SET-086bis-04-pdf

[3] D. K. Barton, *Modern Radar Systems Analysis*. Artech House, 1988. ISBN 0-89006-170-X

[4] L. N. Ridenour, *Radar System Engineering*, 1st ed. New York, McGraw-Hill Book Company, 1947.

[5] M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed. McGraw-Hill Higher Education, 2002. ISBN 9780072881387

[6] D. E. Kerr, et al., *Propagation of Short Radio Waves*. New York, Dover Publications Inc., 1951.

[7] F. E. Nathanson, J. P. Reilly, and M. N. Cohen, "Radar design principles - Signal processing and the environment (2nd revised and enlarged edition)," *NASA STI/Recon Technical Report A*, vol. 91, p. 46747, Jan. 1991.

[8] M. Skolnik, "Fifty years of radar," *Proceedings of the IEEE*, vol. 73, no. 2, pp. 182–197, 1985. doi: 10.1109/PROC.1985.13132

[9] H. Hertz and D. Jones, *Electric Waves: Being Researches on the Propagation of Electric Action with Finite Velocity Through Space*. Macmillan and Co., London and New York, 1893.

[10] H. Griffiths, P. Knott, and W. Koch, "Christian Hülsmeyer: Invention and demonstration of radar, 1904," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 9, pp. 56–60, 2019. doi: 10.1109/MAES.2019.2934814

[11] T. Tyrberg, "Radars barndom i sverige," *TIFF Teknisk Information för Försvarsmaterieltjänsten*, no. 2, pp. 72–77, 2014.

[12] V. Chernyak and I. Y. Immoreev, "A brief history of radar," *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 9, pp. B1–B32, 2009. doi: 10.1109/MAES.2009.5282288

[13] R. I. Wilkinson, "Short survey of japanese radar —I," *Electrical Engineering*, vol. 65, no. 8-9, pp. 370–377, 1946. doi: 10.1109/EE.1946.6434248

[14] R. M. Page, "1977 pioneer award monostatic radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-13, no. 5, pp. 557–571, 1977.

[15] R. M. Trim, "A brief history of the development of radar in Great Britain up to 1945," *Measurement and Control*, vol. 35, pp. 299 – 301, 2002.

[16] R. Watson-Watt, *The Pulse of Radar: The Autobiography of Sir Robert Watson-Watt*. Dial Press, 1959.

[17] H. Boot and J. Randall, "Historical notes on the cavity magnetron," *IEEE Transactions on Electron Devices*, vol. 23, no. 7, pp. 724–729, 1976. doi: 10.1109/T-ED.1976.18476

[18] S. Watts, "The eagle (AN/APQ-7) and its assessment for ASV operation by RAF coastal command," in *2019 International Radar Conference (RADAR)*, 2019. doi: 10.1109/RADAR41533.2019.171326 pp. 1–6.

[19] G. M. Kirkpatrick, "Development of a monopulse radar system," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 2, pp. 807–818, 2009. doi: 10.1109/TAES.2009.5089565

[20] D. K. Barton, "History of monopulse radar in the US," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 3, pp. c1–c16, 2010. doi: 10.1109/MAES.2010.5464419

[21] A. Leonov, "History of monopulse radar in the USSR," *IEEE Aerospace and Electronic Systems Magazine*, vol. 13, no. 5, pp. 7–13, 1998. doi: 10.1109/62.673736

[22] A. Paulraj, R. Roy, and T. Kailath, "Estimation of signal parameters via rotational invariance techniques- Esprit," in *Nineteeth Asilomar Conference on Circuits, Systems and Computers, 1985.*, 1985. doi: 10.1109/ACSSC.1985.671426 pp. 83–89.

[23] C. Cooke, "The early history of pulse compression radar-the history of pulse compression at sperry gyroscope company," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, no. 6, pp. 825–833, 1988. doi: 10.1109/7.18654

[24] M. Chodorow, E. L. Ginzton, I. R. Neilsen, and S. Sonkin, "Design and performance of a high-power pulsed klystron," *Proceedings of the IRE*, vol. 41, no. 11, pp. 1584–1602, 1953. doi: 10.1109/JRPROC.1953.274185

[25] J. Copeland and A. A. Haeff, "The true history of the traveling wave tube," *IEEE Spectrum*, vol. 52, no. 9, pp. 38–43, 2015. doi: 10.1109/MSPEC.2015.7226611

[26] R. H. Varian and S. F. Varian, "A High Frequency Oscillator and Amplifier," *Journal of Applied Physics*, vol. 10, no. 5, pp. 321–327, 1939. doi: 10.1063/1.1707311

[27] V. Gregers-Hansen, "Radar systems trade-offs vacuum electronics vs. solid state," in *Fifth IEEE International Vacuum Electronics Conference (IEEE Cat. No.04EX786)*, 2004. doi: 10.1109/IVELEC.2004.1316173 pp. 12–13.

[28] L. C. Perkins, H. B. Smith, and D. H. Mooney, "The development of airborne pulse Doppler radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 3, pp. 292–303, 1984. doi: 10.1109/TAES.1984.310513

[29] J. Marcum, "A statistical theory of target detection by pulsed radar," *IRE Transactions on Information Theory*, vol. 6, no. 2, pp. 59–267, 1960. doi: 10.1109/TIT.1960.1057560

[30] P. Swerling, "Probability of detection for fluctuating targets," *IRE Transactions on Information Theory*, vol. 6, no. 2, pp. 269–308, April 1960. doi: 10.1109/TIT.1960.1057561

[31] Y. Blanchard, *The origins and development of Radar : an account of the French contribution*, 01 2005, pp. 91–120.

[32] P. Tait, *Introduction to radar target recognition*, ser. IET radar, sonar, navigation, and avionics series. Stevenage UK: The Institution of Engineering and Technology, 2005. ISBN 9780863415012

[33] Y. Blanchard, "A french pre-WWII attempt at air-warning radar: Pierre david's "electromagnetic barrier"," *URSI Radio Science Bulletin*, vol. 2016, no. 358, pp. 18–34, 2016. doi: 10.23919/URSIRSB.2016.7909880

[34] United States Navy. Pacific Fleet and Pacific Ocean Areas, *Japanese Military and Technical Terms*, ser. Cincpac-Cincpoa bulletin, 1945.

[35] L. Gebhard, *Evolution of Naval Radio-electronics and Contributions of the Naval Research Laboratory*, ser. NRL report. [Department of Defense, Navy Department, Office of Naval Research], Naval Research Laboratory, 1979.

[36] S. Broden, "Radar tracking of seaskimmers. Report from field trials with the chase algorithm," in *Radar 97 (Conf. Publ. No. 449)*, 1997. doi: 10.1049/cp:19971775 pp. 743–746.

[37] L. J. Cutrona, W. E. Vivian, E. N. Leith, and G. O. Hall, "A high-resolution radar combat-surveillance system," *IRE Transactions on Military Electronics*, vol. MIL-5, no. 2, pp. 127–131, 1961. doi: 10.1109/IRET-MIL.1961.5008330

[38] C. Kopp, "Russian VHF counter stealth radars proliferate," in *Defence Today*, vol. 8, 2008, pp. 32–36.

[39] "IEEE standard letter designations for radar-frequency bands," *IEEE Std 521-2002 (Revision of IEEE Std 521-1984)*, pp. 1–10, 2003. doi: 10.1109/IEEESTD.2003.94224

[40] M. Martorella and F. Berizzi, "PCL detection fundamentals," *The NATO Science and Technology Organization*, 2018. doi: 10.14339/STO-EN-SET-243-06A-PDF

[41] G. Turin, "An introduction to matched filters," *IRE Transactions on Information Theory*, vol. 6, no. 3, pp. 311–329, 1960. doi: 10.1109/TIT.1960.1057571

[42] H. O. Ramp and E. R. Wingrove, "Principles of pulse compression," *IRE Transactions on Military Electronics*, vol. MIL-5, no. 2, pp. 109–116, 1961. doi: 10.1109/IRET-MIL.1961.5008328

[43] N. Levanon, "Noncoherent pulse compression," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 2, pp. 756–765, April 2006.

[44] J. Kayani and A. Hashmi, "A novel non-coherent radar pulse compression technique based on periodic m-sequences," *Aerospace Science and Technology*, vol. 53, 03 2016. doi: 10.1016/j.ast.2016.03.021

[45] "IEEE standard for definitions of terms for antennas," *IEEE Std 145-2013 (Revision of IEEE Std 145-1993)*, pp. 1–50, 2014. doi: 10.1109/IEEESTD.2014.6758443

[46] J. S. Asvestas, "How far is the far field?" *IEEE Transactions on Electromagnetic Compatibility*, vol. 64, no. 6, pp. 1927–1931, 2022. doi: 10.1109/TEMC.2022.3204787

[47] H. Rohling, "Radar CFAR thresholding in clutter and multiple target situations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-19, no. 4, pp. 608–621, 1983. doi: 10.1109/TAES.1983.309350

[48] J. Berners, *Treatyse on Fysshynge wyth an Angle*. Wynkyn de Worde at Westminster, England, 1496.

[49] W. Radcliffe, *Fishing from the Earliest Times*. John Murray, Albemarle Street, W., London, 1921.

[50] "The authoritative dictionary of IEEE standards terms, seventh edition," *IEEE Std 100-2000*, pp. 1–1362, 2000. doi: 10.1109/IEEESTD.2000.322230

[51] A. W. Doerry, "Catalog of window taper functions for sidelobe control," *Sandia National Lab. (SNL-NM)*, April 2017. doi: 10.2172/1365510

[52] T. T. Taylor, "Design of line-source antennas for narrow beamwidth and low side lobes," *Transactions of the IRE Professional Group on Antennas and Propagation*, vol. 3, no. 1, pp. 16–28, 1955.

[53] P. H. Winston, *Artificial Intelligence*, 3rd ed. Reading, MA, US: Addison-Wesley, 1992. ISBN 978-0-201-53377-4

[54] F. Rosenblatt, "The perceptron - a perceiving and recognizing automaton," Cornell Aeronautical Laboratory, Ithaca, New York, Tech. Rep. 85-460-1, January 1957.

[55] S. Herculano-Houzel, "The human brain in numbers: a linearly scaled-up primate brain," *Frontiers in Human Neuroscience*, vol. 3, 2009. doi: 10.3389/neuro.09.031.2009

[56] W. S. McCulloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, 1943, reprinted in iteAnderson88.

[57] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[58] ——, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, ser. Cornell Aeronautical Laboratory. Report no. VG-1196-G-8. Spartan Books, 1962.

[59] ——, "Perceptron simulation experiments," *Proceedings of the IRE*, vol. 48, no. 3, pp. 301–309, 1960. doi: 10.1109/JRPROC.1960.287598

[60] ——, "Analytic techniques for the study of neural nets," *IEEE Transactions on Applications and Industry*, vol. 83, no. 74, pp. 285–292, 1964. doi: 10.1109/TAI.1964.5407758

[61] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738

[62] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. doi: https://doi.org/10.1016/0893-6080(89)90020-8

[63] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?" *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1223–1247, 2017. doi: 10.1007/s10955-017-1836-5

[64] T. Zaslavsky, *Facing up to arrangements : face-count formulas for partitions of space by hyperplanes / Thomas Zaslavsky*, ser. Memoirs of the American Mathematical Society; no. 154. Providence: American Mathematical Society, 1975. ISBN 0821818546

[65] P. Hinz and S. van de Geer, "A framework for the construction of upper bounds on the number of affine linear regions of relu feed-forward neural networks," arXiv:1806.01918 [stat.ML], 2020.

[66] P. Hinz, "Using activation histograms to bound the number of affine regions in relu feed-forward neural networks," arXiv:2103.17174 [stat.ML], 2021.

[67] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[68] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791

[69] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[70] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[71] K. Chellapilla, S. Puri, and P. Simard, "High Performance Convolutional Neural Networks for Document Processing," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, G. Lorette, Ed., Université de Rennes 1. La Baule (France): Suvisoft, Oct. 2006.

[72] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. AAAI Press, 2011. ISBN 9781577355144 p. 1237–1242.

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[74] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[75] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, San Diego, CA, USA, 2015 arXiv:1412.6980 [cs.LG]*.

[77] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[78] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.

[79] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, jan 2014.

[80] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," arXiv:1602.04938 [cs.LG], 2016.

[81] H. Urkowitz, C. A. Hauer, and J. F. Koval, "Generalized resolution in radar systems," *Proceedings of the IRE*, vol. 50, no. 10, pp. 2093–2105, Oct 1962. doi: 10.1109/JRPROC.1962.288247

[82] U. Peer and N. Levanon, "Compression waveforms for non-coherent radar," in *2007 IEEE Radar Conference*, 2007, pp. 104–109.

[83] A. Seleym, "A new non-coherent pulse compression based on binary codes for on-off keying (ncpc-bc-ook)," in *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, 2013, pp. 731–734.

[84] N. Levanon, I. Cohen, N. Arbel, and A. Zadok, "Non-coherent pulse compression —aperiodic and periodic waveforms," *IET Radar, Sonar Navigation*, vol. 10, no. 1, pp. 216–224, 2016.

[85] J. K. Kayani and A. J. Hashmi, "Comparative study of non-coherent pulse compression and non-coherent pulse integration techniques for radars," in *2013 14th International Radar Symposium (IRS)*, vol. 2, 2013, pp. 696–701.

[86] D. Barton, *Radar System Analysis*, ser. Artech House Radar Library. Artech House, 1979. ISBN 0-89006-043-6

[87] D. Seyfried and J. Schoebel, "Stepped-frequency radar signal processing," *Journal of Applied Geophysics*, vol. 112, 11 2014. doi: 10.1016/j.jappgeo.2014.11.003

[88] P. Grohs, S. Koppensteiner, and M. Rathmair, "Phase retrieval: Uniqueness and stability," 2019, arXiv: 1901.07911 [math.FA].

[89] J. R. Fienup, "Phase retrieval algorithms: a comparison," *Appl. Opt.*, vol. 21, no. 15, pp. 2758–2769, Aug 1982.

[90] R. Beinert and G. Plonka, "Sparse phase retrieval of one-dimensional signals by Prony's method," *Frontiers in Applied Mathematics and Statistics*, vol. 3, p. 5, 2017. doi: 10.3389/fams.2017.00005

[91] Y. Nishizaki, R. Horisaki, K. Kitaguchi, M. Saito, and J. Tanida, "Analysis of non-iterative phase retrieval based on machine learning," *Optical Review*, 01 2020. doi: 10.1007/s10043-019-00574-8

[92] Y. Rivenson, Y. Zhang, H. Günaydın, D. Teng, and A. Ozcan, "Phase recovery and holographic image reconstruction using deep learning in neural networks," *Light: Science Applications, Nature*, vol. 7, no. 2, Feb 2018. doi: 10.1038/lsa.2017.141

[93] C. Metzler, P. Schniter, A. Veeraraghavan, and R. Baraniuk, "prDeep: Robust phase retrieval with a flexible deep network," ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 3501–3510.

[94] Y. Masuyama, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada, "Deep Griffin-Lim iteration," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 61–65.

[95] S. H. Doo, G. Smith, and C. Baker, "Point scatterer modeling of complex targets," in *2014 International Radar Conference*, Oct 2014. doi: 10.1109/RADAR.2014.7060262. ISSN 1097-5764 pp. 1–6.

[96] D. Griffin and Jae Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[97] T. Bendory, R. Beinert, and Y. Eldar, "Fourier phase retrieval: Uniqueness and algorithms," in *Compressed Sensing and its Applications: Second International MATHEON Conference 2015*, H. Boche, G. Caire, R. Calderbank, M. März, G. Kutyniok, and R. Mathar, Eds. Springer International Publishing, 2017, pp. 55–91.

[98] D. Kogan, Y. C. Eldar, and D. Oron, "On the 2d phase retrieval problem," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 1058–1067, Feb 2017. doi: 10.1109/TSP.2016.2631455

[99] R. Balan, P. Casazza, and D. Edidin, "On signal reconstruction without phase," *Applied and Computational Harmonic Analysis*, vol. 20, pp. 345–356, 05 2006. doi: 10.1016/j.acha.2005.07.001

[100] J. K. Kayani, "Performance analysis of noncoherent pulse compression technique," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 4, pp. 2986–2990, 2011.

[101] Y.-B. Ma, "Velocity compensation in stepped frequency radar," Master's thesis, Naval Postgraduate School, Monterey California, 1995.

[102] Y. Liu, H. Meng, Z. Hao, and W. Xiqin, "Motion compensation of moving targets for high range resolution stepped-frequency radar," *Sensors*, vol. 8, 05 2008. doi: 10.3390/s8053429

[103] W. Li, W. Yi, K. C. Teh, and L. Kong, "Adaptive multiframe detection algorithm with range-Doppler-azimuth measurements," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–16, 2022. doi: 10.1109/TGRS.2022.3217128

[104] Y. Barniv, "Dynamic programming solution for detecting dim moving targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-21, no. 1, pp. 144–156, 1985. doi: 10.1109/TAES.1985.310548

[105] X. Li, Y. Yang, Z. Sun, G. Cui, and T. S. Yeo, "Multi-frame integration method for radar detection of weak moving target," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3609–3624, 2021. doi: 10.1109/TVT.2021.3066516

[106] L. R. Moyer, J. Spak, and P. Lamanna, "A multi-dimensional Hough transform-based track-before-detect technique for detecting weak targets in strong clutter backgrounds," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 4, pp. 3062–3068, 2011. doi: 10.1109/TAES.2011.6034689

[107] R. Awadhiya, "Particle filter based track before detect method for space surveillance radars," in *2022 IEEE Radar Conference (RadarConf22)*, 2022. doi: 10.1109/RadarConf2248738.2022.9764213 pp. 1–6.

[108] S. J. Davey and H. X. Gaetjens, *Track-Before-Detect Using Expectation Maximisation*. Singapore: Springer, 2018. ISBN 978-981-13-3971-4

[109] S. J. Davey, M. Rutten, and B. Cheung, "A comparison of detection performance for several track-before-detect algorithms," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, pp. 1–10, 2008.

[110] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[111] E. L. Peters and J. A. Roecker, "Hybrid tracking of low SNR targets," in *2021 IEEE Aerospace Conference (50100)*, 2021. doi: 10.1109/AERO50100.2021.9438272 pp. 1–6.

[112] A. Stroescu, L. Daniel, and M. Gashinova, "Combined object detection and tracking on high resolution radar imagery for autonomous driving using deep neural networks and particle filters," in *2020 IEEE Radar Conference (RadarConf20)*, 2020. doi: 10.1109/RadarConf2043947.2020.9266686 pp. 1–6.

[113] C. Zhu, J. Deng, X. Long, W. Zhang, and W. Yi, "DBU-Net based robust target detection for multi-frame track-before-detect method," in *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*, 2022. doi: 10.1109/ICCAIS56082.2022.9990429 pp. 412–418.

[114] W. Jiang, Y. Ren, Y. Liu, and J. Leng, "Artificial neural networks and deep learning techniques applied to radar target detection: A review," *Electronics*, vol. 11, no. 1, 2022. doi: 10.3390/electronics11010156

[115] M. Roth, "Neural networks for extraction of weak targets in high clutter environments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1210–1217, 1989. doi: 10.1109/21.44038

[116] P. Meiyan, S. Jun, Y. Yuhao, L. Dasheng, and Y. Junpeng, "M-FCN based sea-surface weak target detection," *Journal of Systems Engineering and Electronics*, vol. 32, no. 5, pp. 1111–1118, 2021. doi: 10.23919/JSEE.2021.000095

[117] D. Gusland, S. Rolfsjord, and B. Torvik, "Deep temporal detection - a machine learning approach to multiple-dwell target detection," in *2020 IEEE International Radar Conference (RADAR)*, 2020. doi: 10.1109/RADAR42522.2020.9114828 pp. 203–207.

[118] Z. Baird, M. K. Mcdonald, S. Rajan, and S. J. Lee, "A CNN-LSTM network for augmenting target detection in real maritime wide area surveillance radar data," *IEEE Access*, vol. 8, pp. 179 281–179 294, 2020. doi: 10.1109/ACCESS.2020.3025144

[119] C. Carr, B. Dang, and J. Metcalf, "Radgan: Applying adversarial machine learning to track-before-detect radar," in *2021 IEEE Radar Conference (RadarConf21)*, 2021. doi: 10.1109/RadarConf2147009.2021.9455179 pp. 1–6.

[120] Y. Dai, D. Liu, Q. Hu, and X. Yu, "Radar target detection algorithm using convolutional neural network to process graphically expressed range time series signals," *Sensors*, vol. 22, no. 18, 2022. doi: 10.3390/s22186868

[121] L. Wen, J. Ding, and Z. Xu, "Multiframe detection of sea-surface small target using deep convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–16, 2022. doi: 10.1109/TGRS.2021.3122515

[122] C. Gao, J. Yan, X. Peng, B. Chen, and H. Liu, "Intelligent multiframe detection aided by Doppler information and a deep neural network," *Information Sciences*, vol. 593, pp. 432–448, 2022. doi: https://doi.org/10.1016/j.ins.2022.01.029

[123] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1319–III–1327.

[124] B. Yan, L. Xu, M. Li, and J. Yan, "Track-before-detect algorithm based on dynamic programming for multi-extended-targets detection," *IET Signal Processing*, vol. 11, 03 2017. doi: 10.1049/iet-spr.2016.0582

[125] B. Taha and A. Shoufan, "Machine learning-based drone detection and classification: State-of-the-art in research," *IEEE Access*, vol. 7, pp. 138 669–138 682, 2019. doi: 10.1109/ACCESS.2019.2942944

[126] N. Mohajerin, J. Histon, R. Dizaji, and S. L. Waslander, "Feature extraction and radar track classification for detecting UAVs in civillian airspace," in *2014 IEEE Radar Conference*, 2014. doi: 10.1109/RADAR.2014.6875676 pp. 0674–0679.

[127] X. Zhang, V. Mehta, M. Bolic, and I. Mantegh, "Hybrid AI-enabled method for UAS and bird detection and classification," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020. doi: 10.1109/SMC42975.2020.9282965 pp. 2803–2807.

[128] M. Jahangir and C. Baker, "Persistence surveillance of difficult to detect micro-drones with l-band 3-d holographic radar™," in *2016 CIE International Conference on Radar (RADAR)*, 2016. doi: 10.1109/RADAR.2016.8059282 pp. 1–5.

[129] M. Upadhyay, S. K. Murthy, and A. A. B. Raj, "Intelligent system for real time detection and classification of aerial targets using CNN," in

2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021. doi: 10.1109/ICICCS51141.2021.9432136 pp. 1676–1681.

[130] S. Björklund and N. Wadströmer, "Target detection and classification of small drones by deep learning on radar micro-Doppler," in *2019 International Radar Conference (RADAR)*, 2019. doi: 10.1109/RADAR41533.2019.171294 pp. 1–6.

[131] S. Björklund, "Target detection and classification of small drones by boosting on radar micro-Doppler," in *2018 15th European Radar Conference (EuRAD)*, 2018. doi: 10.23919/EuRAD.2018.8546569 pp. 182–185.

[132] S. Rahman and D. A. Robertson, "Classification of drones and birds using convolutional neural networks applied to radar micro-Doppler spectrogram images," *IET Radar, Sonar & Navigation*, vol. 14, pp. 653–661, May 2020.

[133] D. Raval, E. Hunter, I. Lam, S. Rajan, A. Damini, and B. Balaji, "Detecting drones with radars and convolutional networks based on micro-Doppler signatures," in *2022 IEEE Radar Conference (RadarConf22)*, 2022. doi: 10.1109/RadarConf2248738.2022.9764249 pp. 1–6.

[134] H. Dale, M. Jahangir, C. J. Baker, M. Antoniou, S. Harman, and B. I. Ahmad, "Convolutional neural networks for robust classification of drones," in *2022 IEEE Radar Conference (RadarConf22)*, 2022. doi: 10.1109/RadarConf2248738.2022.9764172 pp. 1–6.

[135] A. Hanewinkel, "Architecture and operational results of feature based automatic radar target classification," in *2019 20th International Radar Symposium (IRS)*, 2019. doi: 10.23919/IRS.2019.8768189 pp. 1–10.

[136] C. Bennett, M. Jahangir, F. Fioranelli, B. I. Ahmad, and J. L. Kernec, "Use of symmetrical peak extraction in drone micro-Doppler classification for staring radar," in *2020 IEEE Radar Conference (RadarConf20)*, 2020. doi: 10.1109/RadarConf2043947.2020.9266702 pp. 1–6.

[137] A. Ramchandran and A. K. Sangaiah, "Chapter 11 - unsupervised anomaly detection for high dimensional data—an exploratory analysis," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, ser. Intelligent Data-Centric Systems, A. K. Sangaiah, M. Sheng, and Z. Zhang, Eds. Academic Press, 2018, pp. 233–251. ISBN 978-0-12-813314-9

[138] G. Shen, Y. Ouyang, and V. Sanchez, "Video anomaly detection via prediction network with enhanced spatio-temporal memory exchange," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022. doi: 10.1109/ICASSP43922.2022.9747376 pp. 3728–3732.

[139] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, oct 2012. doi: 10.1145/2347736.2347755

[140] X. Zhuo, A. Moon, J. Zhang, and S. W. Son, "Cascaded dimension reduction for effective anomaly detection," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021. doi: 10.1109/BigData52589.2021.9671364 pp. 4480–4490.

[141] E. Kaya and G. B. Kaplan, "Neural network based drone recognition techniques with non-coherent S-band radar," in *2021 IEEE Radar Conference (RadarConf21)*, 2021. doi: 10.1109/RadarConf2147009.2021.9455167 pp. 1–6.

[142] H. Dale, C. Baker, M. Antoniou, M. Jahangir, and G. Atkinson, "A comparison of convolutional neural networks for low SNR radar classification of drones," in *2021 IEEE Radar Conference (RadarConf21)*, 2021. doi: 10.1109/RadarConf2147009.2021.9455181 pp. 1–5.

[143] A. Karlsson, "Radar measurements on drones, birds and humans with a 77GHz FMCW sensor." Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5845259

[144] J. Martin and B. Mulgrew, "Analysis of the theoretical radar return signal from aircraft propeller blades," in *IEEE International Conference on Radar*, 1990. doi: 10.1109/RADAR.1990.201091 pp. 569–572.

[145] N. Regev, I. Yoffe, and D. Wulich, "Classification of single and multi propelled miniature drones using multilayer perceptron artificial neural network," in *International Conference on Radar Systems (Radar 2017)*, 2017. doi: 10.1049/cp.2017.0378 pp. 1–5.

[146] Y. Cai, O. Krasnov, and A. Yarovoy, "Simulation of radar micro-Doppler patterns for multi-propeller drones," in *2019 International Radar Conference (RADAR)*, 2019. doi: 10.1109/RADAR41533.2019.171372 pp. 1–5.

[147] L. Lehmann and J. Dall, "Simulation-based approach to classification of airborne drones," in *2020 IEEE Radar Conference (RadarConf20)*, 2020. doi: 10.1109/RadarConf2043947.2020.9266405 pp. 1–6.

[148] C. Bennett, S. Harman, and I. Petrunin, "Realistic simulation of drone micro-Doppler signatures," in *2021 18th European Radar Conference (EuRAD)*, 2022. doi: 10.23919/EuRAD50154.2022.9784488 pp. 114–117.

[149] D. White, M. Jahangir, M. Antoniou, C. Baker, J. Thiyagalingam, S. Harman, and C. Bennett, "Multi-rotor drone micro-Doppler simulation incorporating genuine motor speeds and validation with L-band staring radar," in *2022 IEEE Radar Conference (RadarConf22)*, 2022. doi: 10.1109/RadarConf2248738.2022.9764352 pp. 1–6.

[150] M. Moore, D. A. Robertson, and S. Rahman, "Simulating UAV micro-Doppler using dynamic point clouds," in *2022 IEEE Radar Conference (RadarConf22)*, 2022. doi: 10.1109/RadarConf2248738.2022.9764284 pp. 01–06.

[151] Y. Kim and H. Ling, "Human activity classification based on micro-Doppler signatures using a support vector machine," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 5, pp. 1328–1337, 2009. doi: 10.1109/TGRS.2009.2012849

[152] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[153] D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju, "Why regularized auto-encoders learn sparse representation?" in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 136–144.

[154] W. White, "Low-angle radar tracking in the presence of multipath," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-10, no. 6, pp. 835–852, 1974. doi: 10.1109/TAES.1974.307892

[155] A. Mrstik and P. Smith, "Multipath limitations on low-angle radar tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-14, no. 1, pp. 85–102, 1978. doi: 10.1109/TAES.1978.308582

[156] S. M. Sherman, "Complex indicated angles applied to unresolved radar targets and multipath," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-7, no. 1, pp. 160–170, 1971. doi: 10.1109/TAES.1971.310264

[157] P. Z. Peebles, "Multipath angle error reduction using multiple-target methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-7, no. 6, pp. 1123–1130, 1971. doi: 10.1109/TAES.1971.310213

[158] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986. doi: 10.1109/TAP.1986.1143830

[159] J. E. Evans, J. R. Johnson, and D. F. Sun, "High resolution angular spectrum estimation techniques for terrain scattering analysis and angle of arrival estimation," in *Proceedings of the 1st ASSP Workshop on Spectral Estimation, Hamilton Ont., Canada*, 1981, p. 134–139.

[160] T.-J. Shan, M. Wax, and T. Kailath, "On spatial smoothing of estimation of coherent signals," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 33, pp. 806 – 811, 09 1985. doi: 10.1109/TASSP.1985.1164649

[161] J.-F. Yang and C.-J. Tsai, "A further analysis of decorrelation performance of spatial smoothing technique for real multipath sources," *IEEE Transactions on Signal Processing*, vol. 40, no. 8, pp. 2109–2112, 1992. doi: 10.1109/78.149972

[162] Q. Chen and R. Liu, "On the explanation of spatial smoothing in MUSIC algorithm for coherent sources," in *International Conference on Information Science and Technology*, 2011. doi: 10.1109/ICIST.2011.5765342 pp. 699–702.

[163] M. Viberg, B. Ottersten, and T. Kailath, "Detection and estimation in sensor arrays using weighted subspace fitting," *IEEE Transactions on Signal Processing*, vol. 39, no. 11, pp. 2436–2449, 1991. doi: 10.1109/78.97999

[164] P. Stoica and K. Sharman, "Maximum likelihood methods for direction-of-arrival estimation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 7, pp. 1132–1143, 1990. doi: 10.1109/29.57542

[165] D. Park, E. Yang, S. Ahn, and J. Chun, "Adaptive beamforming for low-angle target tracking under multipath interference," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 4, pp. 2564–2577, 2014. doi: 10.1109/TAES.2014.130185

[166] H. Xiang, B. Chen, M. Yang, T. Yang, and D. Liu, "A novel phase enhancement method for low-angle estimation based on supervised DNN learning," *IEEE Access*, vol. 7, pp. 82 329–82 336, 2019. doi: 10.1109/ACCESS.2019.2924156

[167] H. Xiang, B. Chen, T. Yang, and D. Liu, "Improved de-multipath neural network models with self-paced feature-to-feature learning for DOA estimation in multipath environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5068–5078, 2020. doi: 10.1109/TVT.2020.2977894

[168] V. Karimi, R. Mohseni, and M. Khosravi, "An edge computing framework based on OFDM radar for low grazing angle target tracking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, p. 18, 01 2020. doi: 10.1186/s13638-020-1638-3

[169] C. Eckersten and B.-O. As, "Radar tracking of sea-skimmers, an implementation of 'complex angle'," in *92 International Conference on Radar*, 1992, pp. 46–49.

[170] G. Giunta, L. Lucci, R. Nesti, G. Pelosi, S. Selleri, and F. Serrano, "A comparison between standard and crossfeed monopulse radars in presence of rough sea scattering and ship movements," *International Journal of Antennas and Propagation*, vol. 2010, pp. 1–9, 2010.

[171] M. Agatonovic, Z. Stanković, and B. Milovanović, "High resolution two-dimensional DOA estimation using artificial neural networks," in *2012 6th European Conference on Antennas and Propagation (EUCAP)*, 2012. doi: 10.1109/EuCAP.2012.6206729 pp. 1–5.

[172] M. Wajid, B. Kumar, A. Goel, A. Kumar, and R. Bahl, "Direction of arrival estimation with uniform linear array based on recurrent neural network," in *2019 5th International Conference on Signal Processing, Computing and Control (ISPCC)*, 2019. doi: 10.1109/ISPCC48220.2019.8988399 pp. 361–365.

[173] A. M. Elbir, "DeepMUSIC: Multiple signal classification via deep learning," *IEEE Sensors Letters*, vol. 4, no. 4, pp. 1–4, 2020. doi: 10.1109/LSENS.2020.2980384

[174] Y. Guo, Z. Zhang, Y. Huang, and P. Zhang, "DOA estimation method based on cascaded neural network for two closely spaced sources," *IEEE Signal Processing Letters*, vol. 27, pp. 570–574, 2020. doi: 10.1109/LSP.2020.2984914

[175] G. Papageorgiou, M. Sellathurai, and Y. Eldar, "Deep networks for direction-of-arrival estimation in low SNR," *IEEE Transactions on Signal Processing*, vol. PP, pp. 1–1, 06 2021. doi: 10.1109/TSP.2021.3089927

[176] J. P. Reilly and G. D. Dockery, "Influence of evaporation ducts on radar sea return," *IEE Proceedings F: Radar and Signal Processing*, vol. 137, no. 2, pp. 81–88, Apr. 1990.

[177] R. Paulus, "Evaporation duct effects on sea clutter," *IEEE Transactions on Antennas and Propagation*, vol. 38, no. 11, pp. 1765–1771, 1990. doi: 10.1109/8.102737

[178] J. N. Briggs, *Target Detection by Marine Radar*. The institution of Electrical Engineers, London UK, 2004. ISBN 0 86341 359 5

[179] M. W. Long, *Radar Reflectivity of Land and Sea*. Lexington, Mass. USA: Lexington Books, D.C. Heath and Company, 1975. ISBN 0-669-00050-7

[180] Y. Li and Q. He, "On the ratio of two correlated complex gaussian random variables," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2172–2176, 2019. doi: 10.1109/LCOMM.2019.2941694

[181] Z.-H. Yang and Y.-M. Chu, "On approximating the modified bessel function of the first kind and toader-qi mean," *Journal of Inequalities and Applications*, vol. 2016, no. 1, pp. 1–21, 2016. doi: 10.1186/s13660-016-0988-1

[182] K. A. Wise and D. J. B. Roy, "Agile missile dynamics and control," *Journal of Guidance, Control and Dynamics*, vol. 21, no. 3, 1998. doi: 10.2514/2.4256

[183] H. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988. doi: 10.1109/9.1299

[184] T. Kirubarajan and Y. Bar-Shalom, "Kalman filter versus imm estimator: When do we need the latter?" *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 39, pp. 1452 – 1457, 11 2003. doi: 10.1109/TAES.2003.1261143

[185] F. Lundbäck and V. Wareborn, "Radar target altitude measurement evaluation of different optimization algorithms," Chalmers University of Technology, Gothenburg, Sweden, Tech. Rep., 2013.

[186] C. Yunhe, C. Ting, and W. Shenghua, "Low-angle estimation based on multi-frequency maximum likelihood," in *Proceedings of 3rd International Conference on Multimedia Technology(ICMT-13)*. Atlantis Press, 2013/11. ISBN 978-90-78677-89-5. ISSN 1951-6851 pp. 439–446.

[187] A. S. Lewis and J. Malick, "Alternating projections on manifolds," *Mathematics of Operations Research*, vol. 33, no. 1, pp. 216–234, 2008.

[188] P. Stoica, E. Larsson, and A. Gershman, "The stochastic CRB for array processing: a textbook derivation," *IEEE Signal Processing Letters*, vol. 8, no. 5, pp. 148–150, 2001. doi: 10.1109/97.917699

[189] K. D. Ward, "Compound representation of high resolution sea clutter," *Electronics Letters*, vol. 17, pp. 561–563, Aug. 1981. doi: 10.1049/el:19810394

[190] S. Watts, "Radar detection prediction in k-distributed sea clutter and thermal noise," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-23, no. 1, pp. 40–45, 1987. doi: 10.1109/TAES.1987.313334

[191] A. Parthiban, J. Madhavan, P. Radhakrishna, D. Savitha, and L. Sathish Kumar, "Modeling and simulation of radar sea clutter using k-distribution," in *2004 International Conference on Signal Processing and Communications, 2004. SPCOM '04.*, 2004. doi: 10.1109/SPCOM.2004.1458425 pp. 368–372.

[192] L. Marier, "Correlated k-distributed clutter generation for radar detection and track," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 2, pp. 568–580, 1995. doi: 10.1109/7.381906

[193] BrahMos Aerospace, "Classification of missile," https://www.brahmos.com/content.php?id=10&sid=9, Accessed November 2022.

[194] W. Blair and M. Brandt-Pearce, "Statistics of monopulse measurements of rayleigh targets in the presence of specular and diffuse multipath," in *Proceedings of the 2001 IEEE Radar Conference (Cat. No.01CH37200)*, 2001. doi: 10.1109/NRC.2001.923007 pp. 369–375.

[195] M. Meller and K. Stawiarski, "Robustified estimators of radar elevation angle using a specular multipath model," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 2, pp. 1623–1636, 2020. doi: 10.1109/TAES.2019.2927906