NTNU

Norwegian University of
Science and Technology

ANN based respiration detection using UWB radar

Harald Blehr

Master of Science in Cybernetics and Robotics
Submission date: June 2017
Supervisor: Øyvind Stavdahl, ITK
Co-supervisor: Stefan Werner, IES
Jan Roar Pleym, Novelda AS

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

Master's Thesis

Student's name: Harald Blehr

Field: Engineering Cybernetics

Title (Norwegian): ANN-basert deteksjon av respirasjon ved bruk av UWB radar

Title (English): ANN based respiration detection using UWB radar

Description:
Novelda Ultra Wide Band (UWB) radar is able to detect respiration of humans and animals by measuring mm-differences in distance to the body (e.g. chest wall). This results in a characteristic Doppler spectrum. Algorithms for this purpose exists. One challenge is wrongly classification of oscillating objects (e.g. fans, lamps) with a similar Doppler spectrum. In this project, you will explore the use of artificial neural networks (ANN) to achieve the highest possible sensitivity and specificity for the resulting classification.

Tasks:

1. Study existing literature with relevance for the given problem, and give a brief description of what sub-problems that must be solved in this context.
2. Implement an ANN-based system for classifying respiration from other oscillations. Training data can be generated by expanding existing simulator.
3. Record raw data using Novelda's UWB-radar and verify the performance of the implemented algorithm by recording breathing humans and other oscillating objects, e.g. fans and roof-lamps.

Advisor(s): Øyvind Stavdahl, Assoc. Professor, Dept. of Engineering Cybernetics, NTNU
Jan Roar Pleym, Novelda AS.
Stefan Werner, Professor, Dept. of Electronic Systems, NTNU

Trondheim, 19.01.2017

Øyvind Stavdahl
Supervisor

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Masteroppgave

Studentens navn: Harald Blehr

Fag: Teknisk kybernetikk

Tittel (norsk): ANN-basert deteksjon av respirasjon ved bruk av UWB radar

Tittel (English): ANN based respiration detection using UWB radar

Beskrivelse:
Novelda Ultra Wide Band (UWB) radar er i stand til å detektere respirasjon på mennesker og dyr ved å måle mm-endringer i avstand til kroppen (f.eks. brystkasse). Dette gir seg utslag i et karakteristisk Doppler-spektrum.
Det er implementert algoritmer for dette formål. En utfordring er feilaktig klassifisering av oscillerende objekter (f.eks. vifter, taklamper) med lignende Doppler spektrum.
I denne oppgaven skal du utforske bruk av kunstige nevralnett (ANN) for å oppnå størst mulig sensitivitet og spesifisitet for den resulterende klassifiseringen.

Oppgave

1. Studer tilgjengelig litteratur med relevans for det foreliggende problemet, og gi en sammenfattet beskrivelse av hvilke delproblemer som må løses i forbindelse med dette.
2. Implementere et ANN-basert system for klassifisering av respirasjon vs. annet. Treningsdata for algoritmen kan genereres ved å utvide eksisterende simulator til å inkludere andre objekter.
3. Gjøre rådataopptak vha. Noveldas UWB-radar og verifisere ytelse av implementert algoritme ved å måle mot pustende mennesker og andre oscillerende objekter (vifte, taklamper).

Veileder(e): Førsteamanuensis Øyvind Stavdahl, Inst. for teknisk kybernetikk, NTNU
Jan Roar Pleym, Novelda AS.
Professor Stefan Werner, Inst. for elektronikk og telekommuniksjon, NTNU

Trondheim, 19.01.2017

Øyvind Stavdahl
Faglærer

Abstract

Non-contact detection of human respiration has many possible uses, e.g. health monitoring for clinical institutions, homes or prisons, alarm systems, fire evacuation, industrial or home automation, and triggering of medical imaging. Novelda's Ultra Wide Band radar is able to detect respiration of humans and animals by measuring distance to the chest wall. This results in a characteristic Doppler-spectrum. One challenge is oscillating objects e.g. fans and ceiling lamps, which also give a similar Doppler-spectrum, but for many use-cases should not be treated i.e. classified as human respiration. Artificial Neural Networks (ANN) is a form of Artificial Intelligence (AI) and is an interesting and modern way of solving such classification problems that also seems suitable for this case, where training data is abundant.

This thesis studies the use of Artificial Neural Networks to achieve the highest possible sensitivity and specificity for this classification problem. Throughout the project, a variety of artificial neural networks was tested using Matlab's "Neural Networks Toolbox". The training data was recorded by the author and Novelda employees using Novelda's XeThru UWB radar. The results show that by preprocessing the radar signals into time-vs-frequency images of signal energy, and then use a convolutional neural network as classifier, human respiration can be distinguished from other oscillating objects with a high sensitivity and specificity. The sensitivity and specificity achieved on the test data used for this study was respectively 99.1% and 99.8%, although these results will probably vary with the use of different test data.

The filters of the convolutional layers of the CNNs and recordings that was reverse engineered from a trained CNN was also studied. These studies revealed that the CNN was actually looking for the variations in frequency found in natural human respiration but not in oscillating objects. Also, it only seemed to look for very local patterns, which may be a result of the relatively shallow architectures used here compared to the CNNs used for e.g. face recognition. Thoughts on future work are also discussed in this report. This includes discussions of why deeper CNNs could be suited for this problem, smarter use of some of the tested concepts like e.g. artificial expansion of training data, and the use of ANNs for discovering patterns for use in other types of classifiers.

Acknowledgements

I would like to thank my supervisors from NTNU, Øyvind Stavdahl and Stefan Werner, for very useful discussions on the subject. I would also like to thank my supervisor from Novelda, Jan Roar Pleym, and Novelda employees Ingar Hansen and Magnus Bache, for helping with the recording of training data. It is their personally unique respiration that has shaped the neural connections in the developed ANNs.

List of figures and abbrevations

1 List of Figures

2 List of Abbreviations

ANN Artificial Neural Network

CMOS Complementary Metal–Oxide–Semiconductor

CNN Convolutional Neural Network

DC Direct Current (0 frequency component)

EMD Empirical Mode Decomposition

FCC U.S. Federal Communications Commission

FDR False Discovery Rate

FOR False Omission Rate

FFT Fast Fourier Transform

FN False Negatives

FNR False Negative Rate

FP False Positives

FPR False Positive Rate

FT Fourier Transform

GPU Graphical Processing Unit

HHT Hilbert-Huang Transform

IMF Intrinsic Mode Function

IQ In-phase Quadrature

IR Impulse Radio

LP Low Pass

MSE Mean Squared Error

NPV Negative Predictive Value

NTNU Norwegian University of Science and Technology

PCA Principal Component Analysis

PD Pulse Doppler

PPV Positive Predictive Value

ReLU Rectified Linear Unit

ROC Receiver Operating Characteristic

RF Radio Frequency

RNN Recurrent Neural Networks

RPM Rounds Per Minute

RVSM Radar Vital Signs Monitor

SNR Signal to Noise Ratio

SGD Stochastic Gradient Descent

SoC System on a Chip

SVM Suport Vector Machine

TN True Negatives

TNR True Negative Rate

TP True Positives

TPR True Positive Rate

UWB Ultra Wide Band

VSD Vital Sign Detection

WT Wavelet Transform

Contents

Chapter 1

Introduction

This chapter will cover the background of the project and discuss why it is an interesting field of study. A brief description of the previous work in the field of vital sign detection (VSD) using ultra wide band (UWB) signals will be given. The objectives of this project will be listed, and the outline of the report will be very briefly explained. Throughout the introduction and some of the background theory chapter, introductory and descriptive text will be reused from the author's project report [1] without further mentioning. In the cases where scientific content is reused, it will of course be mentioned. It should also be noted that this project is the author's first encounter with artificial neural networks.

1.1 Background and motivation

Non-contact detection of respiration through clothing and bedding is, just like the heart rate monitoring discussed in the author's project report [1], very useful for intensive care monitoring, long term monitoring and also health monitoring outside of clinical institutions such as homes. Some patients, like infants and burn victims can take damage by contact sensors. Other possible areas of use are security and searching in e.g. firefighting situations [2]. From [3], according to [4], low power IR-UWB is non-ionizing (hence there will be no harm even in continuous monitoring) and has the ability to transmit through obstacles like clothes, bed frame, and blankets.

Novelda's XeThru Ultra Wide Band (UWB) radar is able to detect breathing and in some cases pulse of humans and animals on a distance of a few meters, this is called Vital Sign Detection (VSD). Central in today's signal processing is the use of Pulse Doppler (PD) and Fast Fourier Transform (FFT). One problem with this method is that the sensor may be fooled by non-human oscillating objects, such as oscillating roof-lamps and rotating fans. The use of artificial neural networks for breathing detection is new to Novelda, and will therefore be an interesting field of study that may lead to a smarter detection system.

An example the XeThru radar in use today can be found at Novelda's blog [5]: "XeThru respiration sensors used in Pilot project by Norwegian Police". In Trondheim police district, the XeThru sensors are used to monitor vital signs of inmates. This gives the local police force a better overview of its inmates, and can hopefully prevent overdoses and potentially save lives. This system detects movement and breathing rate.

1.2 Previous work

[6] gives a good overview of previous work on the use of UWB radars for vital sign detection. Only a short summary will be given here. Already in the 1970's, the use of radar for monitoring of human physiologic function was considered [4]. [7], [8] and [9] were the first demonstrating non-invasing sensing with microwave radars. For sensing heart beats [9], however, a cessation in breathing was required. The work on microwave Doppler radar receivers incorporating analog and digital signal processing to separate the weak heart signals and the large breathing signals continued during the 1980's and 1990's with [10], [11], [12], [13] and [14]. [14] developed a Radar Vital Signs Monitor (RVSM) used to detect heart and respiration rate of athletes at the 1996 Olympics in Atlanta.

More recent work [4], [15], [16], [17], [18], [19], [20] and [21] makes use of the UWB radar for detection of heart and respiration rates. According to [6], this work focuses demonstrating the capability and feasibility of the use of UWB radar together with FFT-based signal processing in this field, without seriously discussing or analyzing accuracy or reliability of the methods.

Novelda does not utilize artificial neural networks, but some student work has been done earlier that uses ANNs to e.g. to distinguish walking humans from walking pet animals [22].

1.3 Objectives

The objective of this project was to explore the use of Artificial Neural Networks (ANNs) for classifying respiration from other oscillations in radar signals, and to design and implement an ANN to achieve the best possible sensitivity and specificity for the resulting classification. The objectives in standard NTNU format both in english and norwegian can be found right after the title page.

A study of today's knowledge and results in the field was necessary to get some insight into the field and to find possible architectures and training schemes for the ANN. Novelda did not have any previous neural network based solutions. The sub problems mentioned in the problem description are interpreted to include choosing a framework for exploring ANNs, the preprocessing of the input data, selecting a suitable training algorithm and ANN architecture, and test, evaluate and improve this ANN. The mentioned existing simulator was not used

at all because it had to be modified extensively, and the importance of realistic training data was considered big enough to schedule enough time for recording a variety of real signals that could be used for both training and testing.

1.4 Why Neural Networks

According to [23], designing the right neural network and tuning it can be time-consuming compared with other machine learning techniques such as support vector machines (SVMs). It may give the best performance, but other methods may also work satisfyingly and is often faster to develop. If optimal performance is important, methods that require specialized knowledge, such as neural networks, should be considered. The use of ANNs is explored in this project because it seemed an interesting new approach for Novelda to the problem of distinguishing human respiration from oscillating objects that for many use-cases should be considered as noise.

Other approaches that also could be interesting to explore for the problem discussed in this project, is principal component analysis (PCA), Hilbert–Huang transform (HHT) and other linear classifiers such as support vector machines (SVM). The Hilbert–Huang transform uses empirical mode decomposition (EMD) to decompose dynamic signals into components called intrinsic mode functions (IMF), making it comparable to the Fourier Transform (FT) and the Wavelet Transform (WT).

1.5 Outline

The second chapter covers some very useful background theory for understanding the field of vital sign detection using ultra wide band signals and artificial neural networks. In the third chapter, the methods and the equipment used during the project are explained. The fourth chapter describes the tests executed without giving any numeric results. Selected results of the implemented ANNs are given in chapter five, and discussed in chapter six together with the work method and possible future work. Chapter seven concludes the work, both in terms of results and methods.

Chapter 2

Background Theory

In this chapter, some useful background theory for understanding the field of vital sign detection using ultra wide band signals and artificial neural networks will be covered. It is assumed that the reader has knowledge of basic signal processing and mathematics. In this project, a type of ANN called convolutional neural networks (CNN) was mostly used, and will therefore be emphasized also in this chapter.

2.1 Respiration physiology

This section will briefly describe lung anatomy and the chest motions that Novelda's radar is able to measure.

2.1.1 Lungs anatomy

As thoroughly explained in [24], the lungs are a pair of air-filled organs located in the chest. When air is inhaled, it is conducted through the trachea which divides and enters the two lungs through branches called bronchi, which divides further into smaller and smaller branches ending in small air sacs called alveoli, as illustrated in figure 2.1. Small bloodvessels in the alveoli absorbes oxygen from the inhaled air, and releases carbon dioxide. The lung structure is supported by the pleura which is a thin tissue layer covering the lungs, and the interstitium which is thin cell layers between the alveoli.

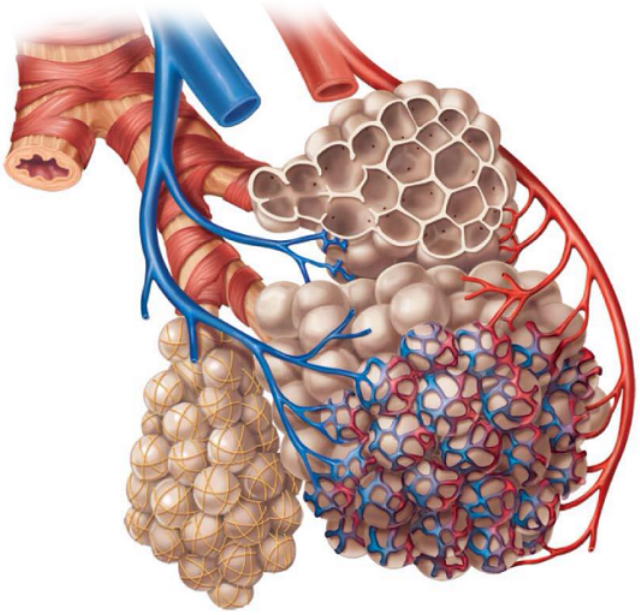

Figure 2.1: Alveolis with small blood vessels. Modified picture from [25]

2.1.2 Chest wall motion

The lung motion caused by inhaling and exhaling propagates through the chest wall, and results in a displacement of the chest surface easily measurable by Novelda's UWB radar.

2.2 Vital Sign Detection using Doppler radar

The chest displacement caused by respiration can be measured from a distance using Doppler radar [26]. This section is taken from the authors project report [1], with some minor changes.

2.2.1 Mathematical model

The radar will measure reflections from stationary reflectors, breathing motion, other irrelevant movement and heartbeats. A simple model for the periodic chest movement caused by respiration or pulse, relative to the radar antennas are [26]:

$$R_m(t) = R_0 + R_h(t) = c\tau_0/2 + c\Delta\tau/2 \cdot sin(\omega_h t), \tag{2.1}$$

where c is the speed of light, τ_0 is travel time, $\Delta\tau$ is maximum deviation in travel time and ω_h is the respiration angular frequency. This model assumes that the target can be modeled as a point scatterer, the radio wave is only reflected from the chest surface. The movement will appear as phase modulation in a single channel receiver, so that the reflected signal can be modeled as:

$$v_R(t) = A_R cos(\omega_c t - \frac{2\omega_c R_m(t)}{c}), \tag{2.2}$$

where ω_t is angular carrier frequency. In the IQ plane, this becomes a complex base band signal:

$$b(t) = A_s e^{i\phi_s} + A_m e^{i\phi_m(t)} + n(t) \tag{2.3}$$

$$A_s e^{i\phi_s} = \sum A_n e^{i\frac{2\omega_c R_n}{c}} \tag{2.4}$$

$$\phi_m(t) = \frac{2\omega_c R_m(t)}{c} = \phi_h(t) + \phi_0 \tag{2.5}$$

where $A_s e^{i\phi_s}$ is the sum of the stationary reflectors in the scene, $A_m e^{i\phi_m(t)}$ is the response from the moving part of the target and $n(t)$ is receiver noise. In the last equation, $\phi_h(t)$ is the time varying phase proportional to $R_h(t)$ and ϕ_0 is a fixed phase corresponding to the fixed distance R_0. Here, only one moving target is assumed. The signal from a single point scatterer is illustrated in figure 2.2.

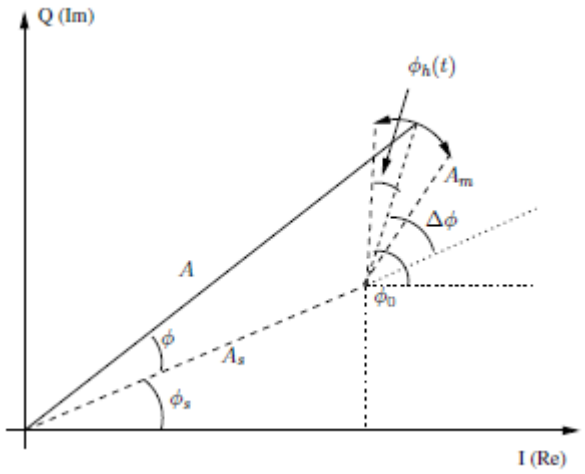

Figure 2.2: Signal from a single point scatterer. Modified figure from [26].

2.2.2 Removing clutter

The chest movement part can be extracted by estimating and removing the static part. The static part can be estimated by creating an estimator for the center of the fractional circle that is the chest movement part, using e.g. some least squares fitting algorithm. Figure 2.3 illustrates a clutter estimate (red) based on the fractional circle signal (blue). Units shown are dergrees in the Real vs. Imaginary plane. It should be noted that removing clutter was not a part of the preprocessing for the final convolutional neural networks in this project. Instead, the mean of each window was simply subtracted before a following Fourier transform.

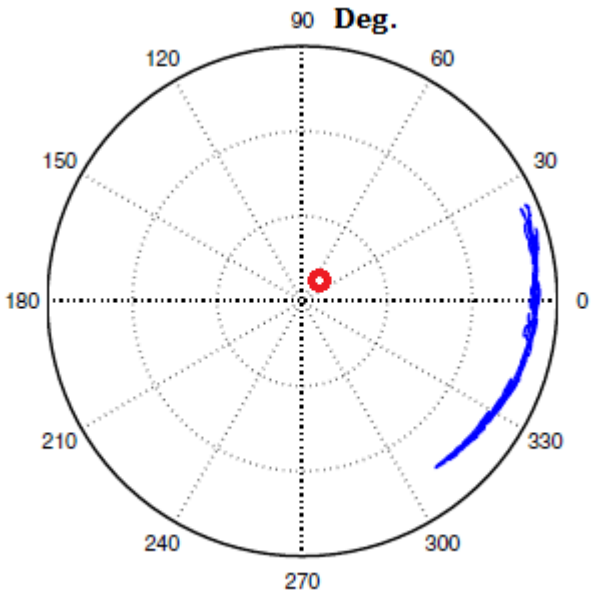

Figure 2.3: Clutter estimate. Modified figure from [26].

2.3 Ultra Wide Band Radar

Ultra wide band (UWB) are signals whose energy are spread over a wide spectrum. In other words, the fractional bandwidth is above some defined limit, typically 20%. This section is also taken from the authors project report [1], with some minor changes.

2.3.1 Background

In 2002, the U.S. Federal Communications commission (FCC) legalized the use of UWB signals, defined by a set of spectral masks [6]. This lead to huge interest in UWB signaling. Pros with UWB compared to narrowband signals are better material penetration, possibility for high data transmission rate and low cost due to need of few analogue parts. Also, its low signal energy per frequency enables it to appear below the noise floor of narrowband signals and hence in many cases not disturb these. Medical UWB applications are now limited to 3.1 to 10.6 GHz [6].

2.3.2 Novelda's Radar

The UWB radar used in this project is described on Novelda's homepage [27]: "Novelda's XeThru Impulse Radar is a complete CMOS radar system integrated on a single chip. This technology is used to implement a high-precision electromagnetic sensor for human vital sign monitoring, personal security, environmental monitoring, industrial/home automation and other novel sensor applications."

2.3.3 In-phase Quadrature demodulation

In-phase Quadrature (IQ) demodulation is used to get access to phase information, and to reduce sampling rate (reduce storage space) by working in the baseband. IQ data describes continuous signals by using two sinusoidal out of phase signals. This is the format of the data as received from Novelda's radar, and the method of IQ demodulation will therefore be described here. The demodulation consists of three steps:

- Down-mixing: The real valued RF-signal is multiplied/mixed with a complex sinusoid signal $x_{IQ}(t) = x_{RF}(t) \cdot e^{-i2\pi f_{Demod}t}$. The resulting signal $x_{IQ}(t)$ is complex, the spectrum is moved down (to the left), and is no longer symmetric about zero.
- LP filtering: To remove the negative frequency spectrum and noise outside the desired bandwidth, the spectrum is now LP-filtered. This removes approximately half the signal energy, so the remaining signal is multiplied with $\sqrt{2}$.
- Decimation: The Nyquist theorem states that the sampling frequency now can be reduced to twice the cutoff frequency of the filter without loss of information. Because we have a complex signal, the bandwidth of the signal equals the complex sampling rate (the complex signal doesn't have an ambiguity between positive and negative frequencies, so both sides of $f = 0$ contributes to the bandwidth).

2.4 Artificial Neural Networks (ANNs)

This section gives an introduction to Artificial Neural Networks, with the concepts used for this project emphasized.

2.4.1 Introduction

Artificial Neural Networks (ANNs) are models used in machine learning to solve classification problems or predict behaviour of systems. These models are inspired by the human brains in the way that they consists of a large number of interconnected artificial neurons. Their connections are adjustable, making the ANN able to learn similar to a biological brain. These artificial neurons are usually organized in layers of neurons, with each layer doing processing at a higher level of abstraction than the previous, resulting in an output at the wanted abstraction level. This is very similar to e.g. the human visual cortices processing information from our eyes. As opposed to our brains, that are tuned by evolution over hundreds of millions of years, ANNs are tuned by using specialized learning algorithms. [23] gives a thorough introduction to neural networks and deep learning.

To create a working ANN, the engineer specifies an architecture for the ANN and an algorithm used for training the ANN. This includes the number of neuron layers, the number of neurons in each layer and how the layers are connected to each others. The neurons contain parameters for weighting their inputs and adding a bias, and also an activation function used to compute their output. The parameters starts as random numbers and are learned during training, but the shape of the activation function is given by the engineer.

2.4.2 Artificial neurons

Artificial neurons are the basic building blocks of the artificial neural network. An artificial neuron is shown in figure 2.4 along with the authors very nice drawing of a biological neuron for comparison. Different activation functions give the neurons different behaviour, the most used versions will be briefly described here.

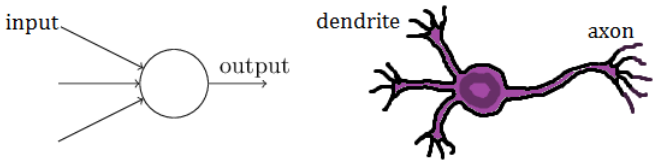

Figure 2.4: A simple neuron model figure. Modified figure from [23].

Perceptrons

These simple neurons have a boolean output, in other word a simple threshold value. They make training very difficult because a tiny adjustment in one of the parameters of one single perceptron may cause its output to flip and cause large and complicated effects in the rest of the network. A continuous function is therefore wanted to simplify training, but perceptrons are great for learning the concept of neural nets. The function is shown below in figure 2.5. The axes are unit-less.

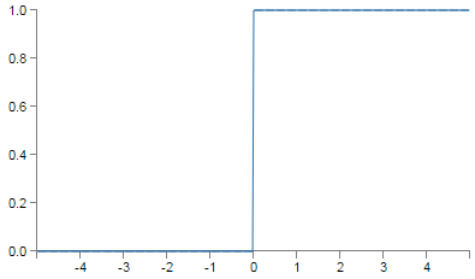

Figure 2.5: Perceptron activation function. Modified figure from [23].

Sigmoid neurons

These are the most used neuron model today. Because a small and gradually change in its parameters only cause a small and gradually change in its output in a linear manner, it is simple to train. The function is shown below and illustrated in figure 2.6. The axes are unit-less. During training, all weights of one neuron must either increase or decrease together. This restriction seems to be a disadvantage, suggesting the use of something like the tanh neurons.

$$\frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{2.6}$$

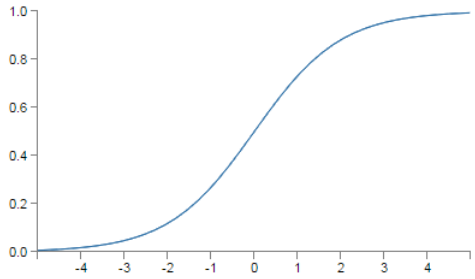

Figure 2.6: Sigmoid activation function. Modified figure from [23].

Tanh neurons

Based on the hyperbolic tangent (tanh) function, these neurons output values span a different interval. The shape of the function is however very close to the Sigmoid neurons, and they are trained using the same techniques. The function is shown below and illustrated in figure 2.7. The axes are unit-less. This function is symmetrical about zero, and accepts both positive and negative activations as opposed to the Sigmoid function.

$$tanh(w \cdot x + b) \tag{2.7}$$

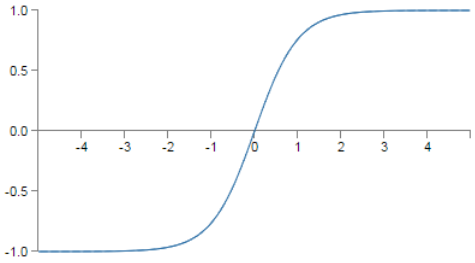

Figure 2.7: Tanh activation function. Modified figure from [23].

Rectified linear neurons/units

Rectified linear units (ReLU) biggest difference from the previously mentioned neurons is that their activation function does never saturate. They are usually trained using the same algorithms, but are in many cases faster to train due to the non-saturating activation function shown below and illustrated in figure 2.8. The axes are unit-less.

$$max(0, w \cdot x + b) \tag{2.8}$$

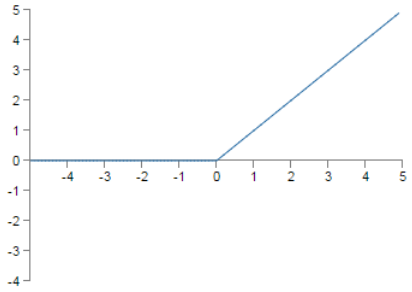

Figure 2.8: ReLU activation function. Modified figure from [23].

The understanding of when and why to use this activation function instead of the tanh or the Sigmoid function is very limited. The ReLU activation function

will be used during most of this project because of the results they provided, in terms of accuracy.

2.4.3 ANN architecture

Artificial neurons are connected together in layers. In the most basic form, each neuron in a layer is connected to every neuron in the previous layer through its input weights, as shown in the figure below. The neurons in the input layer is directly connected to the pixels/values of the input image/vector. This layer is only able to answer very simple questions at pixel-level. The later layers combines these answers into more complex and abstract answers. In classification problems, the output layer usually has the same number of neurons/outputs as there are classes in the problem, and the network will be trained to give the highest output value at the output corresponding to the correct class of its input.

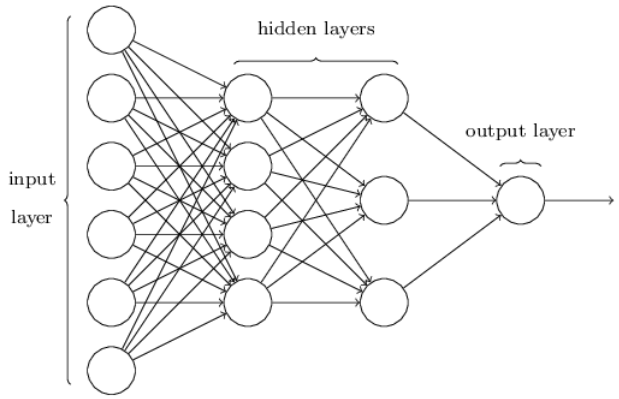

Figure 2.9: A simple feedforward network. Modified figure from [23].

2.4.4 Feedforward and deep networks

Networks with all neurons in a layer connected to all neurons in the previous layer, and no feedback, as in figure 2.9 are called feedforward networks. Networks with two or more hidden layers, as also illustrated in figure 2.9 are called deep networks. Deep feedforward networks in this form will be tested in this project, but the focus will be on convolutional neural networks explained later.

As explained in [23], deep networks make it possible to compute advanced functions with fewer neurons than shallow networks. The main problem with deep networks is the training process, because all parameters must be adjusted based on only the outputs of the very last layer. This gives rise to the unstable gradient problem, among other problems, that are discussed in eg. [23].

2.4.5 Convolutional neural networks (CNNs)

This is a type of feedforward ANN that is inspired by our visual cortex, which is a part of the brain that processes visual information. The neurons of the first layer(s) are not connected to all input pixels, but to a small window called the receptive field, see figure 2.10.

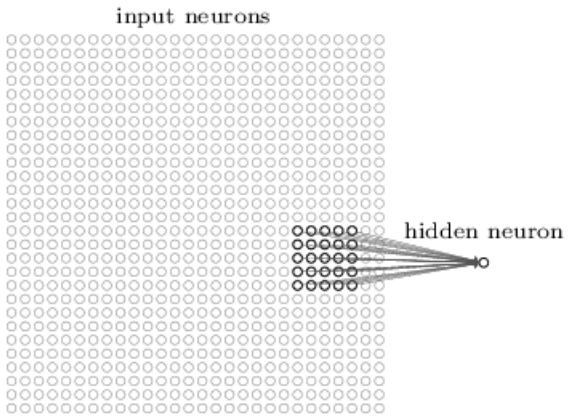

Figure 2.10: Receptive field. Modified figure from [23].

Each layer has multiple channels, and every neuron in a channel shares parameters. This means each channel searches for one specific pattern over the entire input. This makes it possible to save complexity and size by specifically looking only for local spatial patterns. This allows deeper and larger architectures and easier training for the following layers, making these networks very good at classifying images. Figure 2.11 illustrates a typical CNN architecture, with a convolutional layer, a pooling layer and two feedforward layers. The pooling layer is a downsampling of the output of the convolutional layer used to decrease complexity for the rest of the layer. The same algorithms can be used to train CNNs as is used for feedforward ANNs, with slightly modifications.

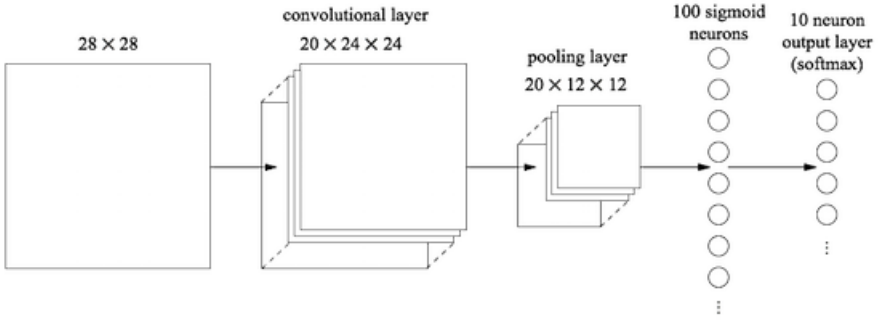

Figure 2.11: A typical CNN architecture. Modified figure from [23].

2.4.6 Other architectures

Other architectures include the Recurrent Neural Networks (RNNs) which contain connections between neurons formed as loops, enabling a form of memory or dynamic behavior. These are used for e.g. speech recognition. This project will be limited to feedforward deep convolutional networks.

2.5 Training ANNs

The most used way of training ANNs is to use a version of the gradient descent algorithm to optimize a cost function that is a function of the parameters, i.e. weights and biases of the ANN.

2.5.1 Cost functions

A cost function (sometimes called loss function) is used during training to compute how close the ANNs estimate is to the correct answer. As opposed to classification accuracy, these functions provide a smooth output with gradients that can be used to adjust the parameters of the ANN. Mean squared error (MSE) can be used as cost function when training ANNs:

$$C(w,b) = \frac{1}{2n}\sum_x \|y(x) - a\|^2, \tag{2.9}$$

where w and b are the weights and biases of the ANN, n is number of trainig samples, y is a vector with 1 at the position corresponding to the correct class and a is the output of the ANN.

The cross-entropy cost function is much used as a cost function because of its ability to make the ANN learn a lot faster from the start than MSE when

weights and biases are initialized very badly. The cross entropy cost function is shown below.

$$C(w,b) = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)] \tag{2.10}$$

For this project, an activation function called the Softmax function was used for output layer of neurons. The output of neuron j then becomes

$$\frac{\exp(z_j)}{\sum_k \exp(z_k)}, \tag{2.11}$$

$$z = w \cdot x + b \tag{2.12}$$

This function's output values are positive and always sum to 1, and can therefore be thought of as a probability distribution, useful for classification problems. A cost function called the log-likelihood function combined with the Softmax output layer gives the same behaviour as the cross entropy function combined with Sigmoid neurons as output, but with the advantage of the probability distribution compatible output.

2.5.2 Gradient descent

Solving the optimization problem of minimizing the cost function analytically is impossible due to the vast number of parameters for ANNs and the complicated ways these depend on each others. The gradient descent algorithm computes the gradient of the cost function when changing the parameters of the ANN slightly:

$$\nabla C = \left(\frac{\delta C}{\delta p_1}, ..., \frac{\delta C}{\delta p_n}\right), \tag{2.13}$$

where C is the output of the cost function and v are the parameters of the ANN. The parameters are changed in the opposite direction to optimize the cost function:

$$\Delta v = -\eta \cdot \nabla C, \tag{2.14}$$

where v is a vector of the parameters of the ANN, η functions as a learning rate and ∇C is the gradient of the cost function. This is done iteratively.

The gradient descent algorithm is often used on the mean values of random batches of input samples called mini-batches to increase speed. This is called stochastic gradient descent (SGD). In this project, an expansion of the SGD called momentum is used. This is a term added to the function above containing weighted previous steps. This enables the learning to go faster and avoid local minima, but with a risk of overshooting at the global minimum. When all training samples are used, one epoch of training is done. For the following epochs, new random mini-batches are selected from the training samples.

2.5.3 Back-propagation

The gradient of the cost function with respect to any weight or bias is calculated using the back-propagation algorithm. This algorithm feeds the ANN with an input vector/image and the output of the ANN is computed. The error values for the output neurons are computed using the cost function. At last, the error values are propagated backwards through the ANN, resulting in an error value for each weight and bias. These error values indicate the contribution of its corresponding parameter to the output.

2.5.4 Regularization

A major problem associated with ANNs are overfitting. This means that the ANN is able to memorize much of the input data without being able to understand the important patterns and generalize to new situations. This problem increases with the complexity of the ANN and the lack of sufficient amounts of training data. An example from this project of an overfitting ANN is shown in figure 2.12.

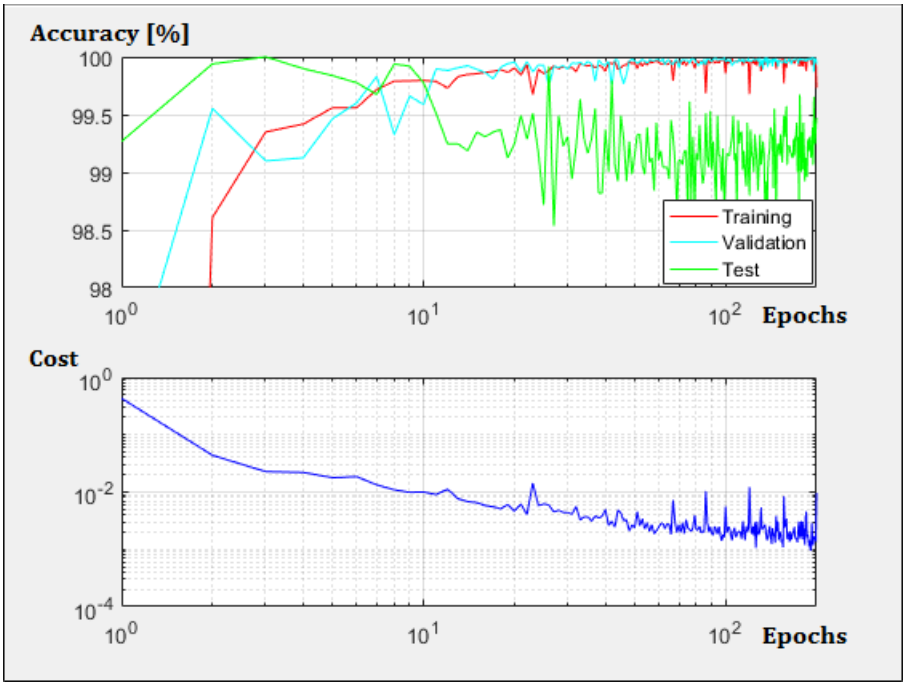

Figure 2.12: An overfitting ANN's accuracy and cost during training.

Even though the accuracy on the training and validation data continues to increase and the output of the cost function shown below continues to decrease,

the accuracy on the test data reaches a top and then starts to decrease. This is because the ANN starts to learn patterns in the samples of the training data that is not general to the class of those samples. Regularization techniques addresses the problem of overfitting, and can also in some cases improve accuracy and provide more stable and replicable results.

The regularization methods used in this project was L2 regularization, dropout layers, and artificial expansion of the training data. These will be described here.

2.5.5 L2 Regularization

L2 regularization, also known as weight decay, is a technique where a term is added to the cost function to prevent parameters to grow very large. As discussed in [23], this may lead to a few very large parameter values that make changes in other parameters almost insignificant. The L2 regularization term is shown below, it is simply the sum of the squared parameter values.

$$\frac{\lambda}{2n}\sum_{w} w^2 \tag{2.15}$$

If cross-entropy is used together with L2 regularization, the cost function becomes:

$$C(w,b) = -\frac{1}{n}\sum_{x}[y\ln a + (1-y)\ln(1-a)] + \frac{\lambda}{2n}\sum_{w} w^2 \tag{2.16}$$

2.5.6 Dropout layers

Dropout layers are layers that disconnects neurons with a given probability, normally 0.5. This forces the present neurons to learn more robust features that works even when only a random subset of the other neurons are present, because the neurons cant rely on the presence of all other neurons. It is important to note that this mechanism is used only during training. This can be thought of as training many different ANNs and averaging the effects of the different ANNs. If they overfit in different ways, the averaged result will be less overfitting.

2.5.7 Artificial Expansion

Artificial expansion (also called algorithmic expansion) of the training data means generating more training data by shifting, rotating, adding noise to, or in other ways modify the existing training data. More varied training data helps the network generalize to new situations better. An example of a smart and successful way of implementing artificial expansion is [28], where the objective was to recognize handwritten digits. They used a form of artificial expansion they called elastic distortions, where they used knowledge of the natural oscillations in hand muscles to generate realistic variations of their training data.

This study will also try to use knowledge of the signals of Novelda's UWB radar to expand the available training data in a realistic way.

2.5.8 Hyper-parameters

These parameters are global for the ANN, and include the following factors:

- The number of epochs to train
- Mini-batch size
- Learning rate
- Regularization factors
- Momentum factor

Finding a set of hyper-parameters that generates an ANN that works any better than a noise generator in the first place can be difficult. When a set of parameters are found that enables the ANN to learn, they can be optimized. The hyper-parameter space has many dimensions, the hyper-parameters have dynamic effects on each other, and the time needed to train long enough to get any useful results can be large. This makes optimizing hyper-parameters difficult and possibly very time consuming. Tuning parameters one by one, when a usable set is found, can give a good feeling of how the parameters work. A very simple form of grid-search was also used for this project, looping through variations of two parameters at a time.

2.5.9 Matlab for training ANNs

Matlab offers a toolbox called "Neural Network Toolbox" that implements a framework for training ANNs. This is the framework used for this entire project.

2.6 Classification

This section will describe the statistical metrics used to evaluate the ANNs developed during this project.

2.6.1 Confusion matrices

Confusion matrices are a type of performance measurement, an example from this project is shown in figure 2.13, where the red and green sections constitute a confusion matrix. It gives more information than the accuracy, which was used during development to make design choices. It shows true positives (TP) top left, false positives (FP) top right, false negatives (FN) bottom left and true negatives (TN) bottom right. For this project, positives are respiration and negatives are noise or other oscillating objects. The overall accuracy can be found in the blue field at the bottom right in the same figure.

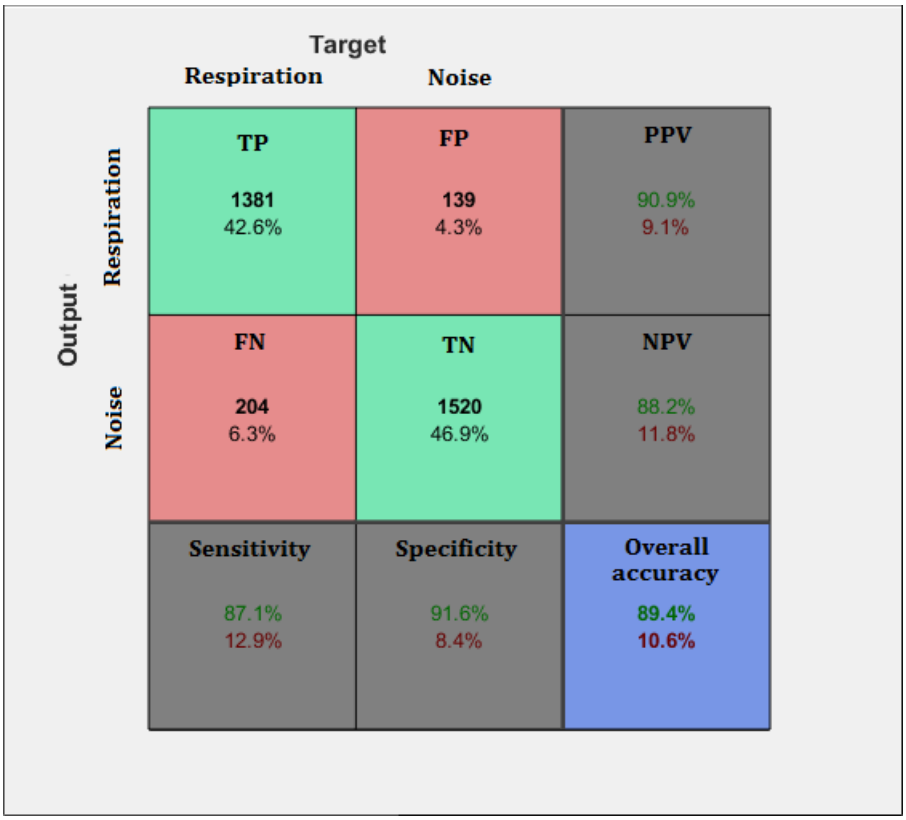

Figure 2.13: Example confusion matrix.

2.6.2 Sensitivity and specificity

One of the main objectives of this project was to achieve the highest possible sensitivity (also called probability of detection or True Positive Rate (TPR)) and specificity (also called True Negative Rate (TNR)). These statistical measures are calculated from the values of the confusion matrix:

$$sensitivity = \frac{TP}{TP + FN} \tag{2.17}$$

$$specificity = \frac{TN}{TN + FP} \tag{2.18}$$

In figure 2.13, the row at the bottom shows sensitivity at the left and specificity at the middle, both in green text. The red text below shows respectively False Negative Rate (FNR) and False Positive Rate (FPR), in other words miss rate and probability of false alarms.

2.6.3 Positive and negative predictive values

Another interesting perspective on the values of the confusion matrix is positive and negative predictive values (PPV and NPV). These numbers indicate the accuracy or the performance of the classification. The PPV is the probability that a frame or time interval classified as containing respiration truly contains respiration. Similar, the NPV is the probability that a frame classified as containing only noise truly contains only noise. The PPV and NPV are defined as:

$$PPV = \frac{TP}{TP + FP} \tag{2.19}$$

$$NPV = \frac{TN}{TN + FN} \tag{2.20}$$

In figure 2.13, the column at the far right shows positive predictive value at the top and negative predictive value at the middle. The red text below shows respectively False Discovery Rate (FDR) and False Omission Rate (FOR).

2.6.4 Reciever operating characteristic (ROC)

For classification problems in biometrics (as is the case of this study), medicine, psychology and other fields, it is interesting to study the trade off between sensitivity and specificity when varying the discrimination threshold of the classifier. This is because the cost of the consequences of false positives and false negatives can be very different. When plotting sensitivity against False Positive Rate (FPR) (1 - specificity) for different discrimination threshold values, this trade off becomes clearly visualized. This is called a ROC plot, an example from this project is shown in figure 2.14. The two lines show the mentioned trade-off for both respiration and noise. These will always be exactly opposite to each others, and often only one of them is shown.

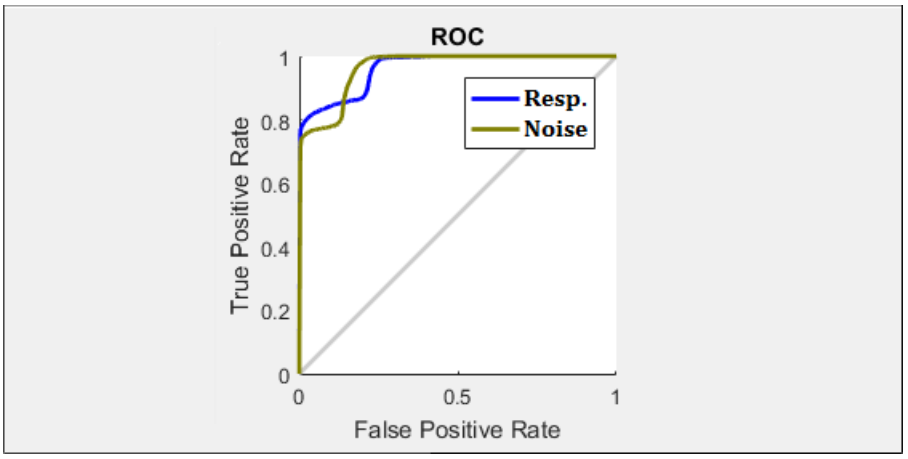

Figure 2.14: Example ROC.

Chapter 3

Methods And Equipment

3.1 Work planning

For the project scheduling, a Gantt chart was constructed to be able to assign roughly enough time for each planned major task. Exploring neural networks demands trying and failing. It was not given from the start exactly what tests should be done. The chart was therefore changed slightly during the project, e.g. time was rescheduled from fully connected ANNs to convolutional ANNs because the first results obtained from the convolutional ANNs seemed very promising. Because this project was the authors first encounter with ANNs, some adjustments of the schedule was expected from the start. An early version of the Gantt chart is shown in figure 3.1.

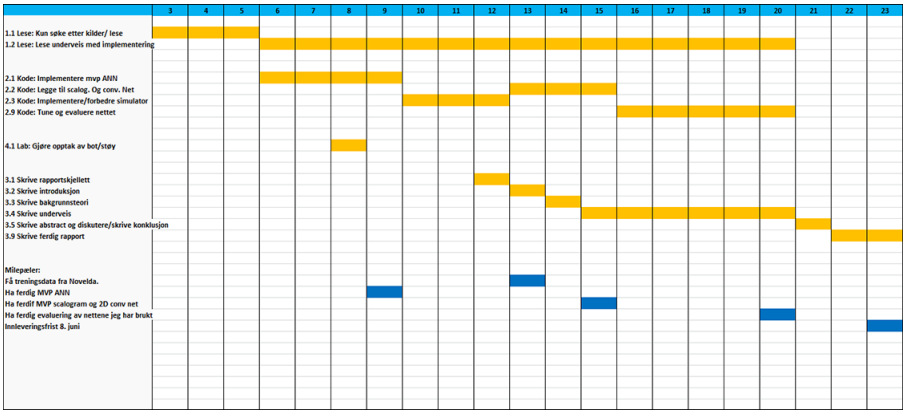

Figure 3.1: Gantt chart.

3.2 Hardware used

3.2.1 GPU

For the training of the ANNs, a GPU owned by the author was used to be able to complete a variety of parameter combinations within reasonably time. Specificcally the nVidia GeForce GTX 960.

3.2.2 Radar and setup

The radar used was Novelda's Xethru UWB radar, shown in figure 3.2. During recording of training and test data, the radar was mounted on a camera tripod. See the Radar setup section in the next chapter for the setup of the radar.

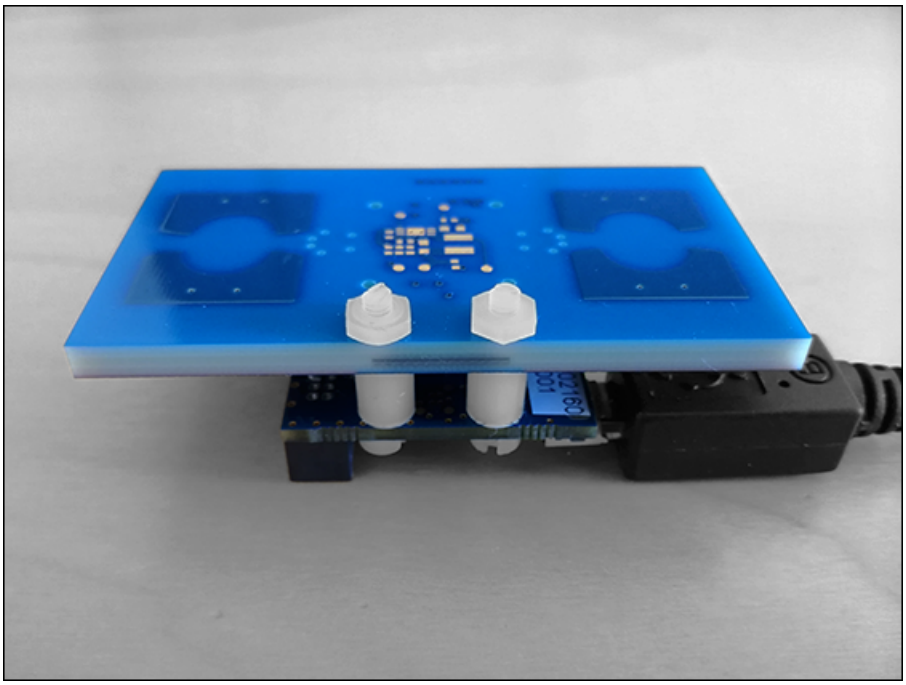

Figure 3.2: Novelda's radar.

3.3 Software used

3.3.1 Xethru Explorer

The recording was done with the radar connected via USB to a laptop, controlled by the user interface provided by Xethru explorer. A screenshot of the

application during recording is shown in figure 3.3

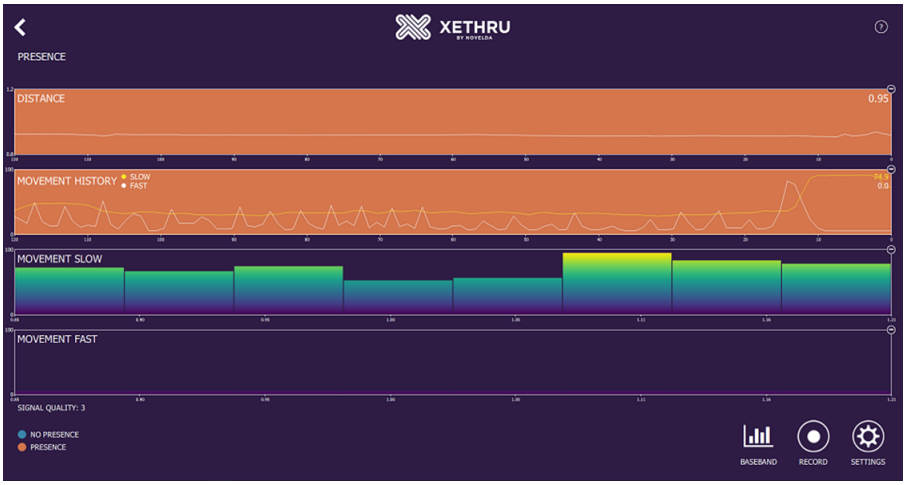

Figure 3.3: XtExplorer during recording.

3.3.2 Matlab

For this project, some different programming languages and packages for ANNs were considered. Matlab was chosen because it provides all functionality needed for this project, and because both the author and Novelda was already used to this environment.

3.4 Design choices

3.4.1 Validation and test data

Before training, the data set was divided into three data sets: Training set, validation and test set. Matlab uses partitions of the training data to calculate accuracy after each epoch. The validation set was not included in the training process, but was used during training to measure classification accuracy. The test data introduced recording situations not used for training at all. This gave a measurement of how well the ANN generalized to new data, and revealed overfitting to the training/validation data. Further, to prevent possible overfitting of the network to the test data when adjusting the architecture or the hyper-parameters, and to give an even better measurement of generalization, the ANN was finally tested on a second test set. In other words, the first test set can be thought of as a validation set for setting the hyper-parameters. This approach is called the "hold out" method. This is how the different architectures was compared in this project, and is illustrated in figure 3.4.

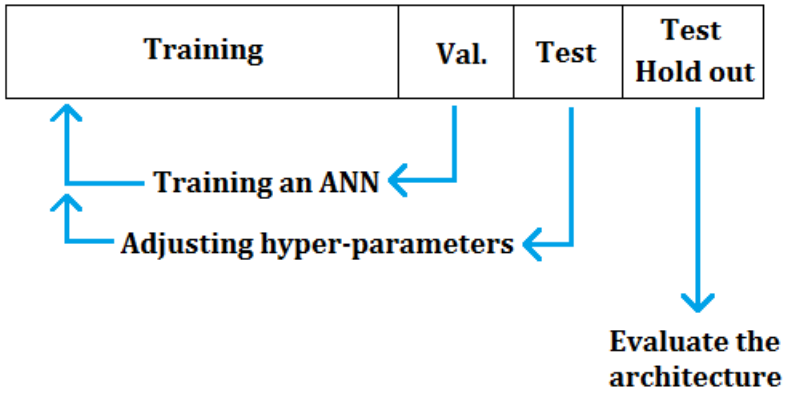

Figure 3.4: Training approach.

3.4.2 Preprocessing

During the project, previously acquired knowledge of Novelda's radar and its signals came in very useful for the preprocessing of the input data for the ANN. An example of this is that scalogram-like images were used as input at a stage in this project because this format had previously shown [1] to contain information that seems relevant for this project with a high level of SNR. This preprocessing may cause a simpler ANN to be able to make good classifications because the ANN no longer needs to learn how to extract frequency information or similar information because this already will be done by the preprocessing.

3.4.3 ANN architecture and training

Architecture and training scheme choices for the ANN were made based on knowledge acquired during the literature research in this project. To be able to make good decisions, a script were made that loops through some values for one or more hyper-parameter at a time, and plots and saves relevant information from each case in a structured manner. This script and its generated plots took some different forms throughout the project as experience and knowledge of what to look for improved.

Chapter 4

Exploring the Artificial Neural Networks (ANNs)

This chapter describes the concepts that were tested during the project, but does mentions few numeric results. A few times, major changes was done leading to results in a new format, so that the following results were not directly comparable to the earlier. Examples of this is changing the training set or the type of information on which the comparisons were based. The next chapter contains selected results to illustrate the main points of what was learned. The code created during this project will not be shown in this chapter, but selected parts of it will be added in appendix A.

4.1 Recording training data

ANNs need training data. This was acquired by recording different situations with Novelda's UWB radar.

4.1.1 The situations described

Signals containing respiration was recorded by the Novelda employees Jan Roar Pleym, Magnus Bache and Ingar Hansen. They recorded themselves at sleep, lying awake, sitting, and finally making small normal movements during the previously mentioned situations.

Signals containing noise was recorded by the author. These recordings contain rotating fans at different angles and rotating speeds, oscillating roof lamps at different maximum angles, an oscillating wall-mounted guitar, and a LEGO bot fitted with motors oscillating a Christmas tree balls at different sizes and oscillating frequencies. The Christmas tree balls were used because of their reflecting capabilities.

4.1.2 Radar setup

During recording of training and test data, the radar was mounted on a camera tripod. Figure 4.1 illustrates a situation where the radar measures the movements of a Lego robot at the top, a sitting person at the bottom left, and a sleeping person at the bottom right. As the figure indicates, a distance of approximately 1 meter was used for the recordings used in this project. Some variations of this distance was a natural result of the different recording situations, different people recording, and the fact that in some situations (e.g. sleeping) the target sometime moves. Figure 4.2 shows pictures of two example setups. A couch to the left and chairs araund Novelda's lunch table to the right.

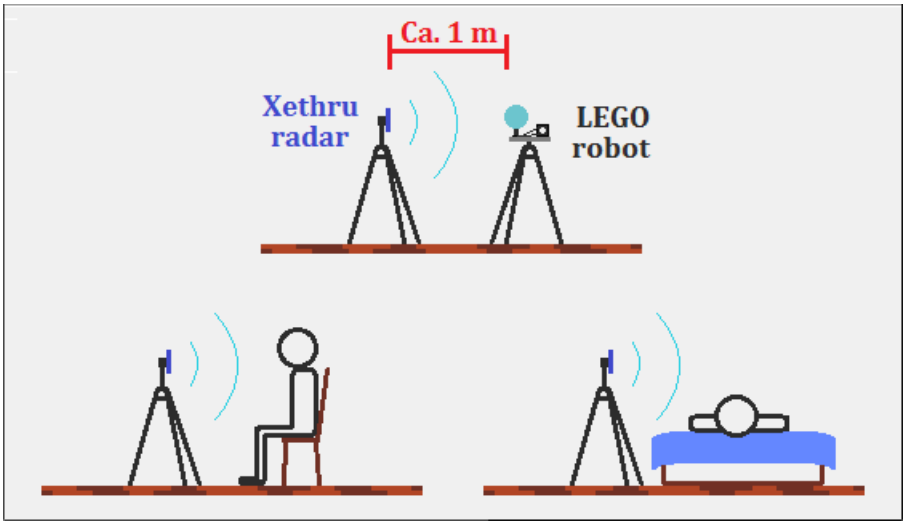

Figure 4.1: Different radar setups with tripod.

Figure 4.2: Examples of radar setups.

4.2 Preprocessing

Preprocessing was done to provide the ANN with relevant information to make classification decisions in a compact and available format. The compact form is important to reduce input size and thereby reducing the size and computational complexity of the ANN. The available form may result in a simpler and shallower ANN because the information needed is more directly available already at the first layer.

Preprocessing may cause loss of information, in other words it may contain non-invertible transformations. This was also considered when choosing the methods for preprocessing in this project. A sufficiently sampled Fourier transform (as will be used in this project) is an example of an invertible transformation, and therefore keeps all information (when rounding errors are considered insignificantly small). This section will describe the preprocessing of the input samples for the ANNs.

4.2.1 The raw data described

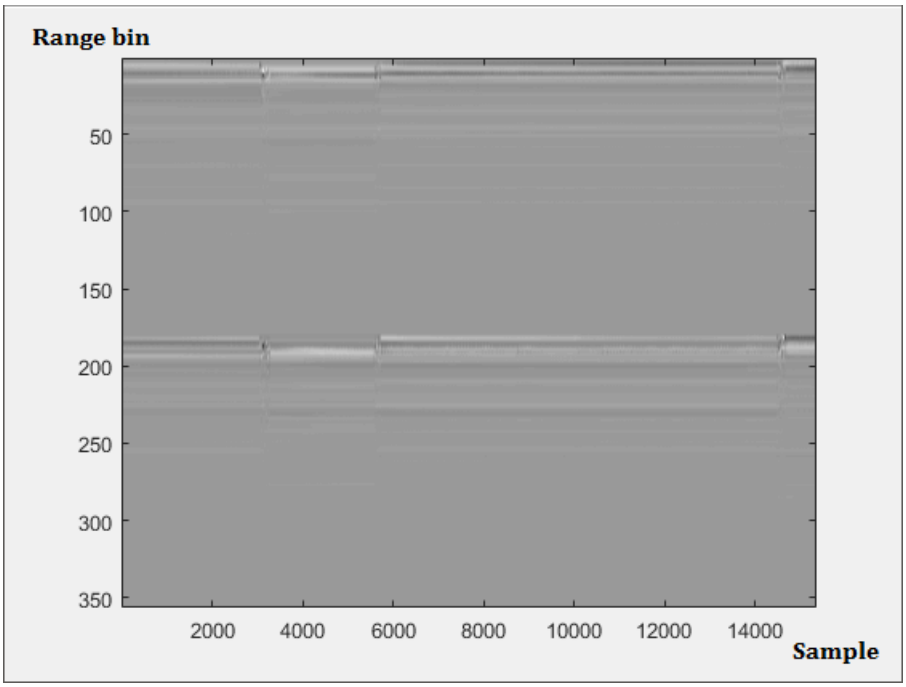

Figure 4.3: A frame matrix containing radar signals.

The raw data was given as .dat files containing recordings for 15 minutes each. When loaded with a function written by Novelda, the data is loaded into two

Matlab matrices. A frame matrix shown in figure 4.3 containes the radar signals, one column for each sample, and one row for each range bin. In-phase and Quadrature channels are stacked vertically, the top half is the In-phase signals. The horizontal lines that can be spotted in the figure corresponds to the range bins that contain reflections. When zooming in as in figure 4.4, the lines are clearly periodic in intensity, which corresponds to an oscillating target, in this case Jan Roar Pleym's chest when sleeping. Another matrix contains information like sample numbers and the length in meters of each range bin.

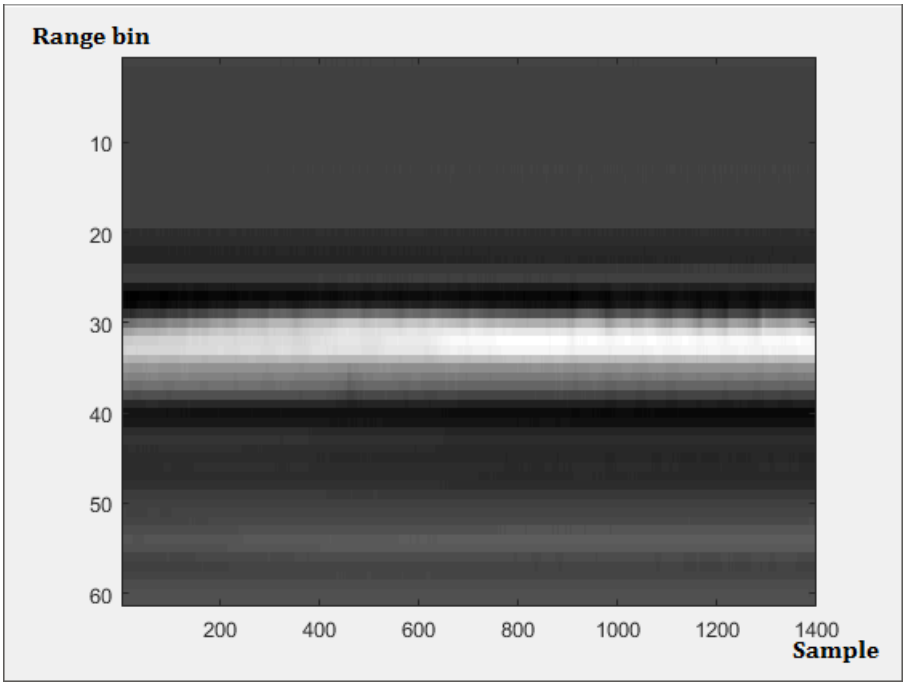

Figure 4.4: Zoomed in on a section of the frame matrix.

4.2.2 Forming a sample

Range gating

The first step of the preprocessing is range gating. In this project, this was done by eyeballing the first part of a recording situation, and then manually programming the correct distance to range gate. This was a fast and simple approach that worked great for this project. An algorithm that outputs the estimated range for the same type of raw data is in development at Novelda, but was not finished at the time of this study. After range gating, the signal is a complex 1D signal, an example is shown in figure 4.5. By zooming in, the signal is clearly oscillating in both channels, as can be seen in figure 4.6, forming a complex signal oscillating in phase around a point given by stationary clutter, as in figure 4.7 and explained in the background theory chapter. At this stage, windows of 20 seconds was extracted from the complex signal, with a distance of 1 second between each start point i.e. overlapping.

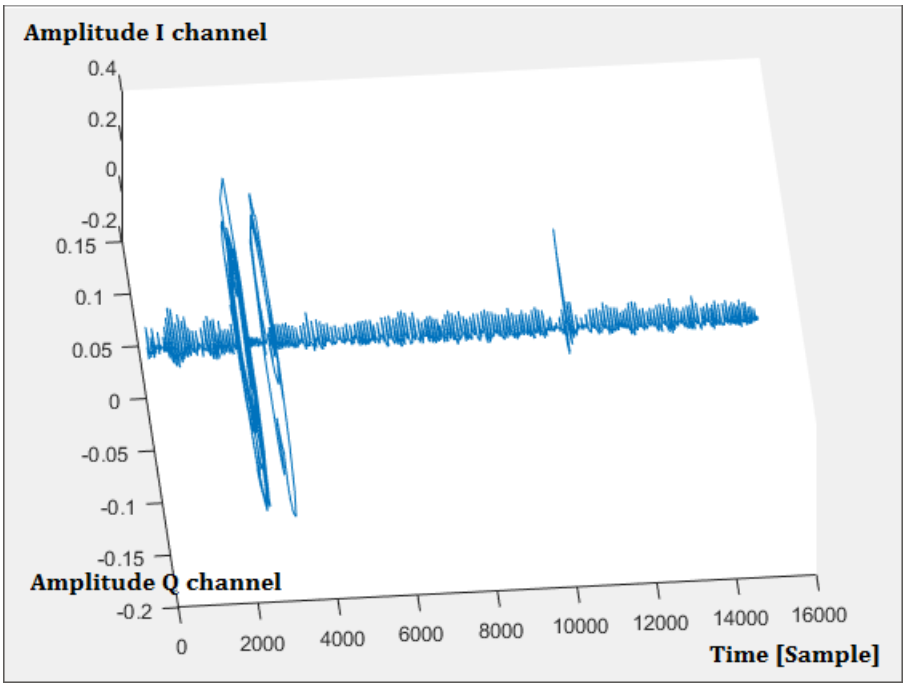

Figure 4.5: Rangegated signal.

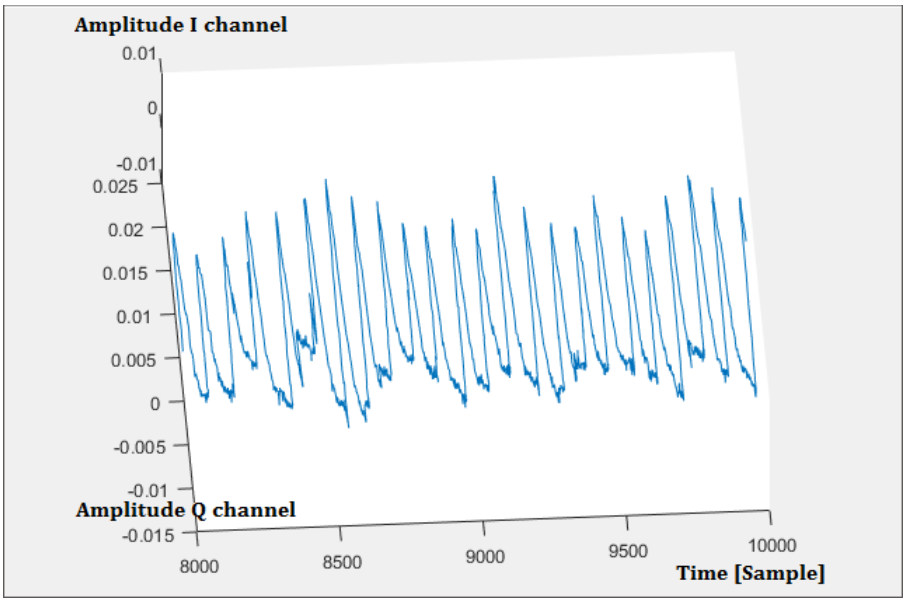

Figure 4.6: Section of rangegated signal.

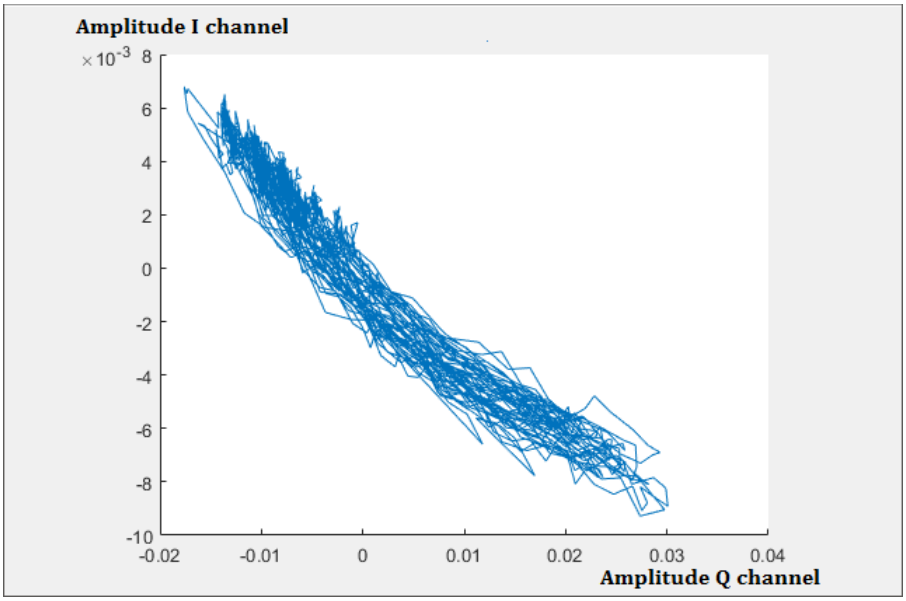

Figure 4.7: Section of rangegated signal, I channel against Q channel.

Input for convolutional ANNs

Each window was Fourier transformed into vertical vectors. Each Fourier transform vector was then stacked horizontally to form time-against-frequency images. These images were downsampled into 50 by 50 pixel images, an example is shown in figure 4.8. Larger values give lighter colors. This choice of resolution was based on looking at what resolutions the assumed relevant information was clearly visible. At this resolution, small changes in oscillating frequency was still visible, and some variation of sleeping respiration frequency was visible in each image. Each image overlapped its neighbour image with all but one column/sample. This gave a large number of varied images to use for training.

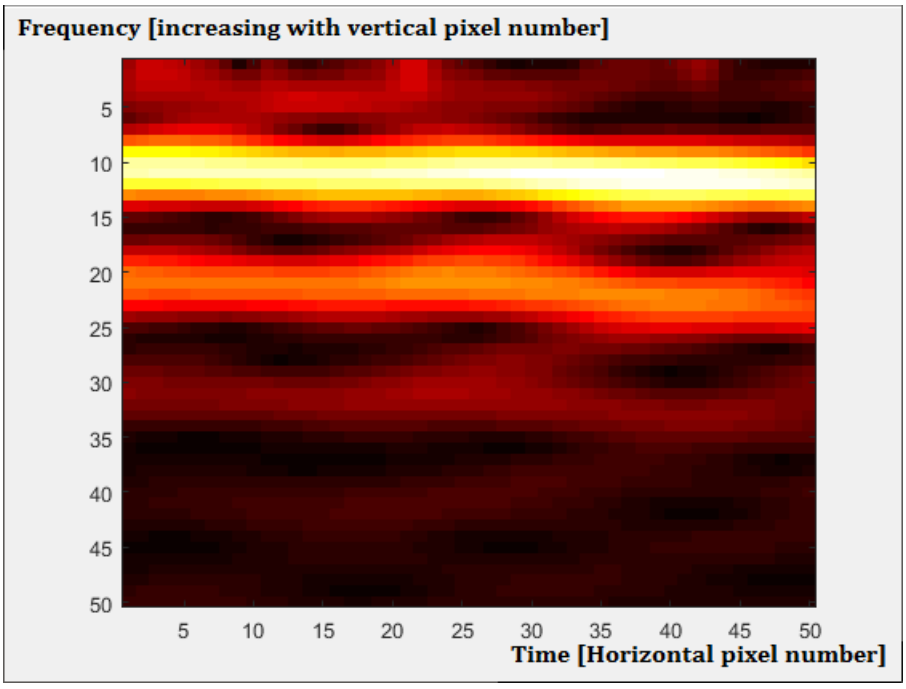

Figure 4.8: Example of an input image.

4.2.3 Managing the samples

The samples were balanced so that the number of samples for each class (respiration and noise/other oscillating reflectors) were the same. This gave a balanced training of the ANN and also intuitive test results. The samples were shuffled and divided into three batches, one for training, one for validation and one for testing.

4.2.4 Artificial expansion

Artificial expansion were used on the training data (not on the test data) to increase the variety of the training data, in the form of a custom made function. For the 2D samples, some different transformations were tested. The function translated the input images along the frequency axis, weakened the signals, added noise and flipped the images. The number of samples was multiplied by approximately 20 after this step.

4.2.5 The sample set described

At this point, approximately 120 000 training samples as illustrated in figure 4.9, 5 000 validation samples and 27 000 test samples was made for the sample set used in all comparisons described in the results chapter. These samples contain processed recordings of all Novelda employees mentioned earlier when sleeping, and also all noise situations mentioned earlier. See the section about Recording earlier in this chapter for reviewing the details.

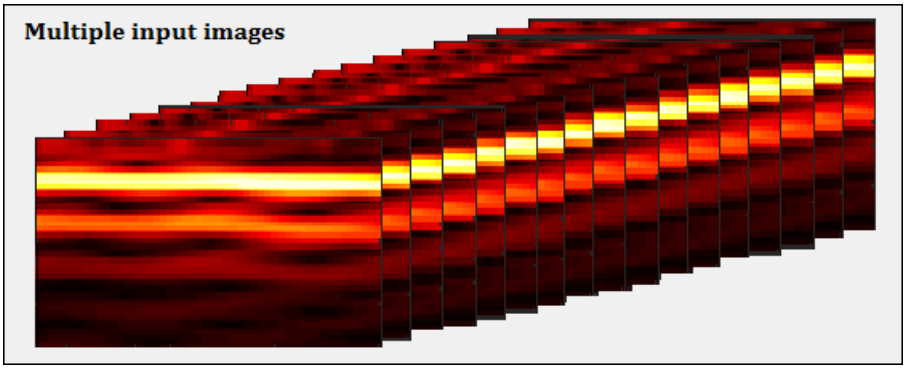

Figure 4.9: Input samples.

4.3 Feedforward neural networks

Fully connected feedforward ANNs were to a small extend explored because it is the simplest way of connecting artificial networks. Very little knowledge of the problem can be utilized, and the ANN serves mainly as a black box. However, it was a natural start before moving on to convolutional neural networks, described in the next section.

4.4 Convolutional neural networks (CNNs)

Creating images that contain both time and frequency information seemed logical because this information is assumed to be useful in distinguishing respiration from other oscillating objects. It then seemed very logical to explore the use of convolutional ANNs on this input, because samples close to each others in time, and frequency content close in frequency are related in patterns, and should therefore be treated differently than samples/frequency content far apart. This way, previously acquired knowledge could possibly be used to increase performance of the classification solution.

4.4.1 Initial architecture

First, some architectures with different number of layers and number of neurons for these layers were tested. One architecture that worked after some trial and error with hyper-parameters, was the one illustrated in figure 4.10. The reason for testing this simple architecture in the first place was mostly based on hunches. It seemed logical to use a convolutional layer because the local patterns in the input image was assumed to contain valuable information. It also seemed logical to start with only one standard feedforward layer because it was assumed that there was no need to look for very complicated compositions of these patterns. Simple networks also generally overfits less than complicated networks. A pooling layer was added because it simplifies the ANN, and because the exact respiration frequency should not matter for the classification problem in this project.

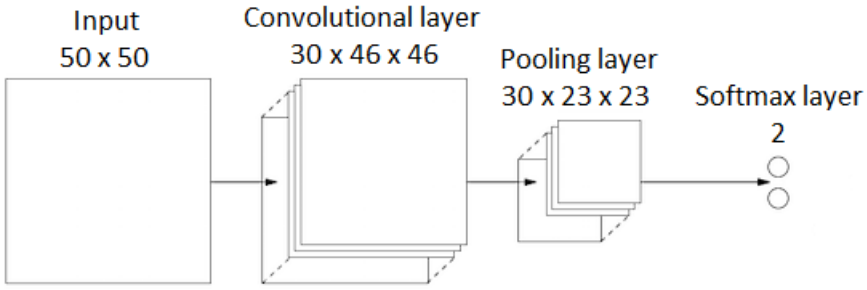

Figure 4.10: A convolutional architecture. Modified illustration from [23].

This is an ANN with one convolutional layer, one pooling layer and one softmax layer. Figure 4.10 shows the shape and size of the input and the outputs of each layer. The number of channels in the convolutional layer (30) was chosen this large as a start to prevent the ANN to be held back by not being able to look for enough different shapes. The filter sizes (5 x 5) was chosen to be able to contain a meaningful section of the input image, e.g. a small section of a

line or a bend. The size of the pooling was chosen to be the same as one of the examples in [23] as a first try. The last layer is the output layer and has the same number of neurons as classes in the classification problem. It was chosen to be a Softmax layer to give output values that can be interpreted as probabilities, which is appropriate for this project.

This architecture was chosen as basis for further study. It was trained for 1000 epochs, which with the training set and training batch size used, corresponds to 1 000 000 iterations of the training algorithm. Figure 4.11 shows the accuracy after each iteration. Its accuracy seemed to almost saturate after much fewer epochs, although it is not easily visually verifyed by this plot, which looks very noisy due to the number of iterations. 40 epochs was therefor chosen for further comparisons of rough adjusting of this architecture to save time. It achieved an accuracy of 91.44% on the test data after 40 epochs.

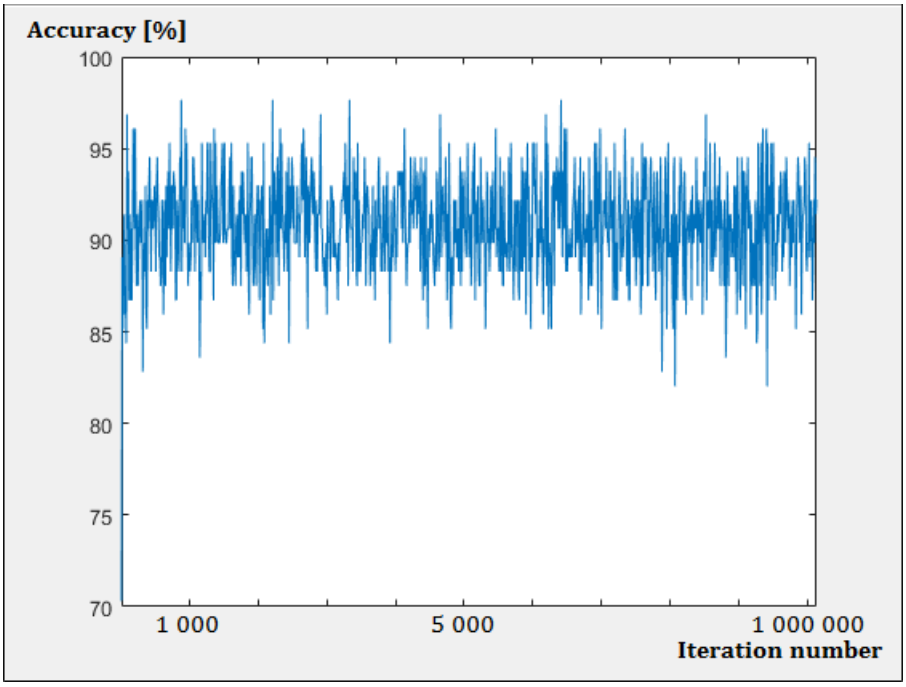

Figure 4.11: Accuracy after each iteration.

4.4.2 Improving the architecture

In order to improve the accuracy, parameters were adjusted one by one, as described in chapter 3.4.3. The following paragraphs describes what parameters were adjusted and the results of these tests.

Number of filters

As shown if figure 4.12, a few of the filters seemed to take the shape of approximately horizontal lines, very much like those visible in the input images. Many filters still looked random. It was assumed that these filters only served as random number generators for the rest of the network, and the number of filters was therefor decreased to 15 for further tests.

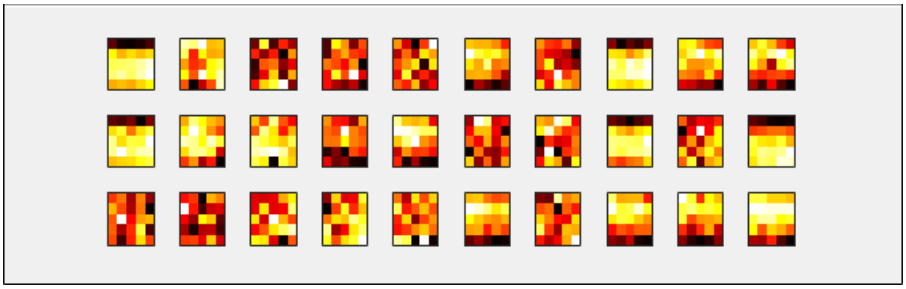

Figure 4.12: Several seemingly random filters.

Filter sizes

A number of different filter sizes were tested, 15x15 pixels gave the best results in terms of accuracy on the test data. An accuracy of 93.02 % was achieved at this stage. A montage of selected filters created during different tests are shown in figure 4.13. It should be noted that only about 5 to 10 filters with such clear structure was trained for each CNN, the rest of the filters were very noisy or contained seemingly only random values. This montage still gives an impression of what sort of patterns the CNNs tends to look for.

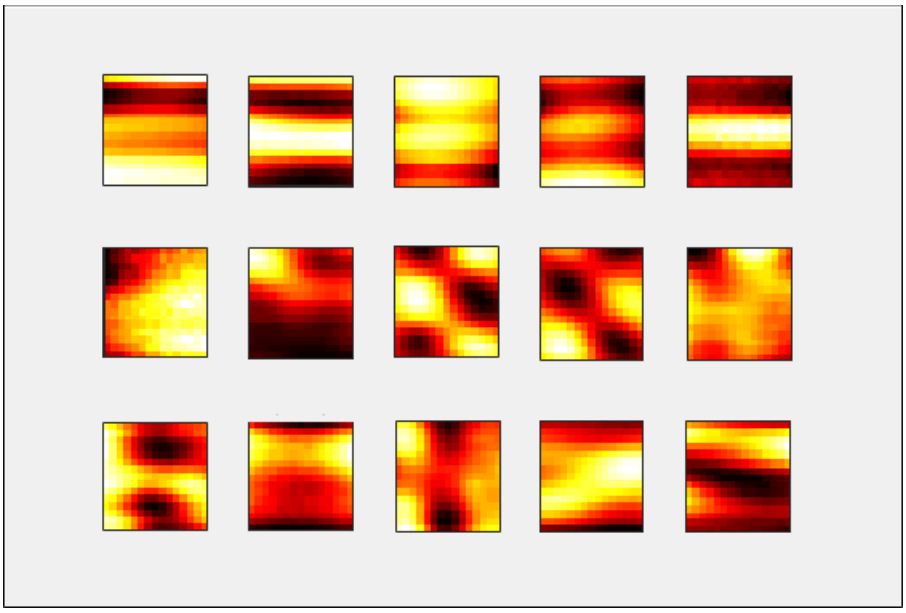

Figure 4.13: Selected 15 x 15 pixels filters.

Pooling sizes

Pooling sizes of 2x2 that were non-overlapping was very close in accuracy to no pooling at all. Larger quadratic pooling sizes reduced accuracy considerably. Because of the simplification a pooling layer introduces, a pooling layer with 2x2 pixel windows was kept for further tests.

A specialized pooling layer

As previously mentioned, it was assumed that the patterns containing the relevant information for the classification problem of this project should be the same for different respiration frequencies. Large rectangular pooling windows which compressed information from the convolutional layer in the frequency dimension was also tested. The best results from these test was achieved with a pooling window of $6x1$ neurons. Figure 4.14 tries to illustrate a similar compression, but with smaller layers for simplicity. An accuracy of 93.48% on the test data was achieved using this pooling layer, actually better that what was achieved with less compression. It is, however, a small change, and could just be coincidental.

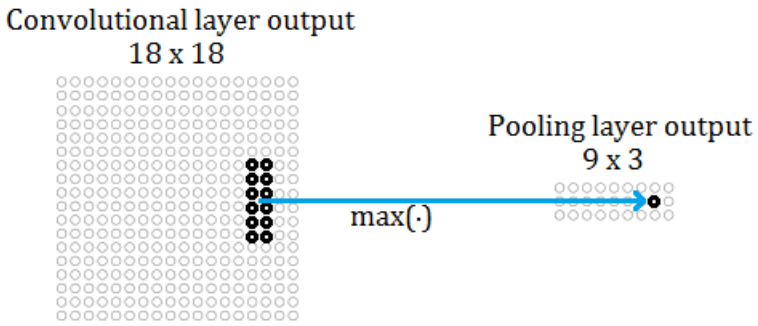

Figure 4.14: Specialized pooling layer. Modified illustration from [23].

4.4.3 Hyper-parameters

Accuracy may also be improved by using a better training schedule or by making modifications to the training algorithm. The following paragraphs describes what parameters were adjusted and the results of these tests.

Initial learn rate

An initial learn rate of 0.02 was found to give the best results in terms of accuracy on the test data, and was used for all tests for this architecture. This factor have no physical unit, and does not make much sense on its own, or if using a different training setup. It is simply mentioned here for completeness, and to point out that it was actually tuned during this project.

Learn rate drop period and factor

By using a training schedule that periodically decreases the learning rate, more precise learning or in other words a fine tuning at the end of the training period can be achieved. This did not improve the accuracy for this architecture.

L2 regularization and momentum

By default, Matlab uses an L2 regularization parameter close to 0 and a momentum factor of 0.9. Changing these did not improve the accuracy for this CNN.

4.4.4 A deeper CNN

Even better accuracies were achieved by using a deeper CNN. By adding another hidden layer of ReLU neurons before the output layer, the CNN should be able

to discover more abstract patterns in the input images. After a lot of trying and failing, a deeper CNN was successfully trained using the dropout technique explained in the background theory chapter.

4.4.5 Architecture

An illustration of this deeper architecture is given in figure 4.15.

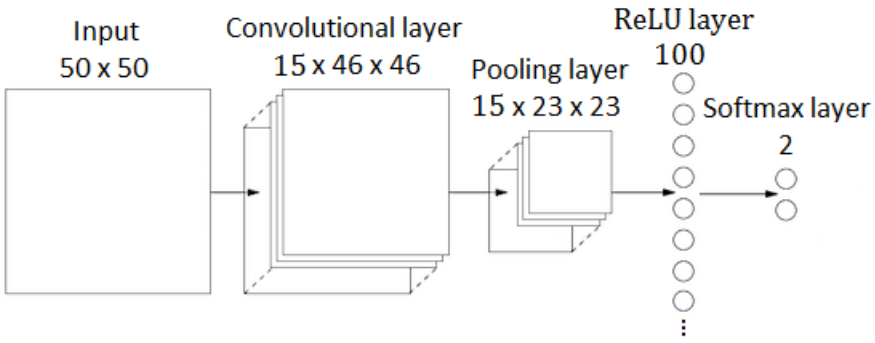

Figure 4.15: A larger convolutional architecture. Modified illustration from [23].

4.4.6 Studying the CNN filters

The filters of the first convolutional layer shows what patterns of pixels the CNN actually looks for. Studying these filters gave valuable insight in what image features is used for the classification in the later layers.

An example is shown in figure 4.16. The numbers above each filter is the mean squared values of the filter pixels. These numbers indicate to what extent the filters have taken the shape of some feature. Low values mean that the filter pixels does not affect the classification result much, and therefore have stayed close to stationary during training. All filters with a relatively high mean square value seemed to have taken the shape of straight horizontal lines, very similar to what can typically be found in recordings of dead objects oscillating with a stationary, even frequency. As a result of this observation, it was discovered that many of the noise recordings were done in a way creating radar signals of very good SNR relative to typical recordings of unknowing humans. The preprocessing algorithm was therefore extended to normalize the signals before the Fourier transforming. This greatly improved the CNNs ability to classify new recordings done with different setups.

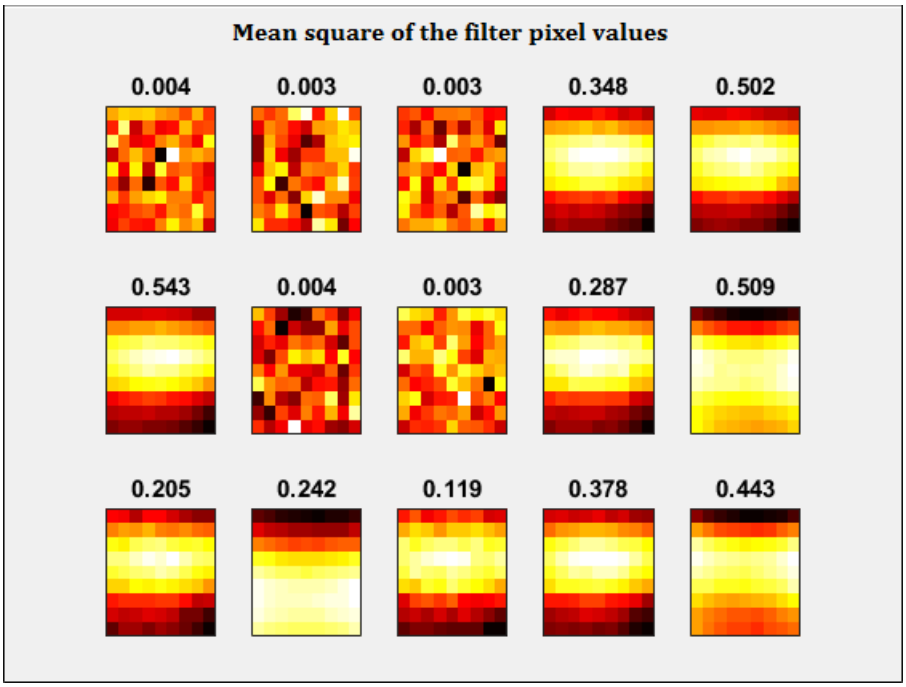

Figure 4.16: Horizontal line-like filters with mean square pixel values.

Another example is shown in figure 4.17. Many of these filters seems to have taken the form of the edges of horizontal lines. Based on this observation, it was discovered that a large part of the training images containing noise contained sudden starts and stops of the noisy oscillation, hence the clear ending and starting lines. This was due to the way the training data was recorded. More samples containing continuous noise was added, and this considerably improved the ability of the later CNNs to generalize to a lot of new noisy situations.

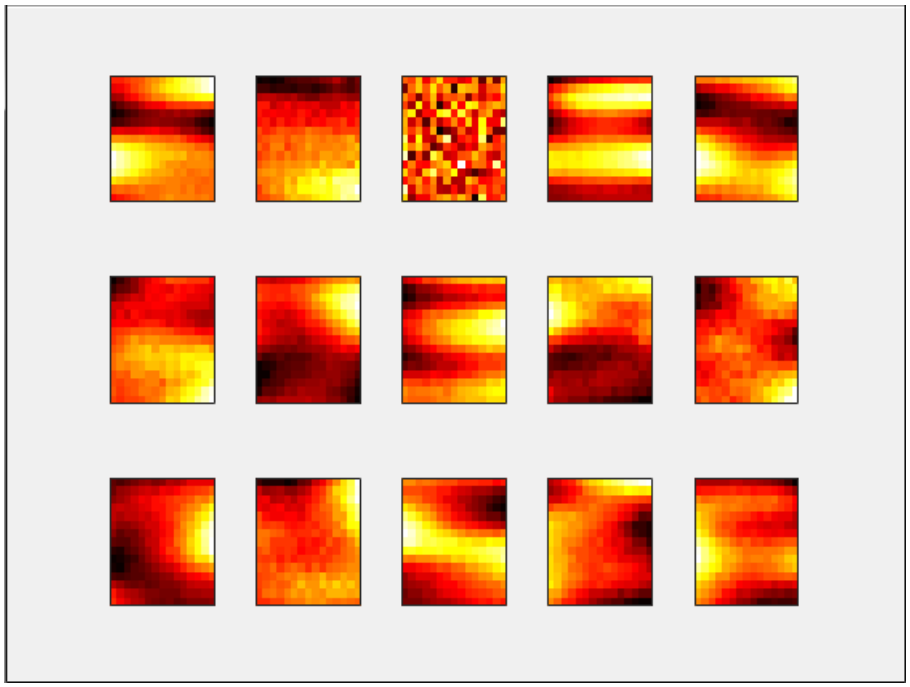

Figure 4.17: Horizontal line edge-like filters.

4.5 Respiration as seen by a CNN

Getting trained CNNs to tell what images looks the most like respiration and noise recordings was attempted because this kind of reverse engineering was assumed to give valuable information on what the CNN actually was using as basis for its classification, but in a different way than studying the filters of its first layer. The way the CNN was given the ability to tell what a respiration-like image should look like, was simply by iteratively adding Gaussian white noise to a random initialized image, and compute the gradient of the CNNs output when fed with this image. This gradient could be used for adding noise to the image in a way that made the image more and more respiration-like each iteration. One of the first results is shown in figure 4.18. The color mapping is different from the rest of the figures simply because the figure was made in an early stage of the project.

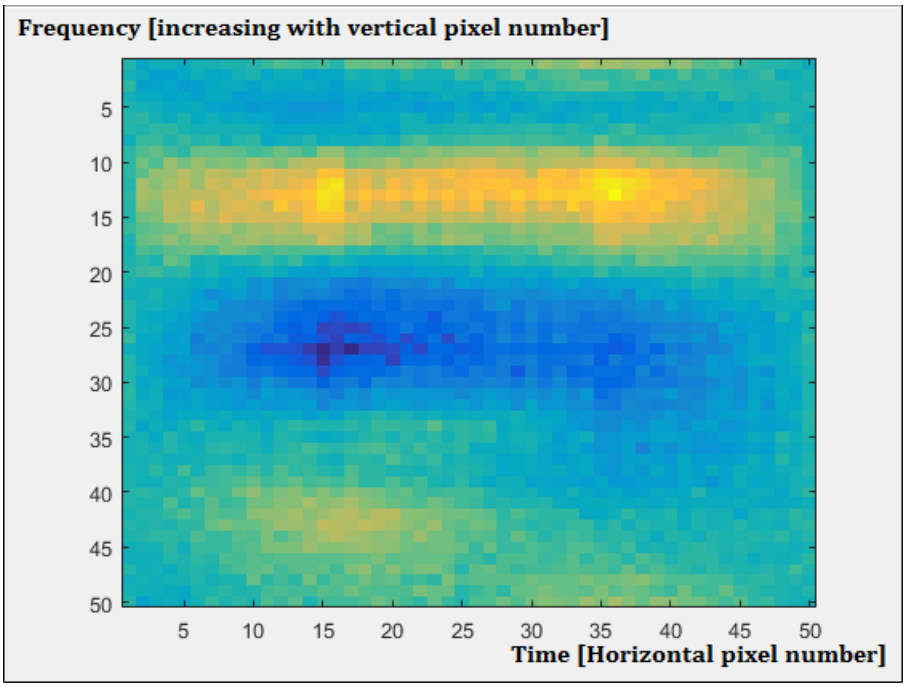

Figure 4.18: One of the first reverse engeneered images.

It clearly shows a broad horizontal ridge in the earea of about 10 to 17 RPM, which is typical respiration frequencies. Based on this observation, it was discovered that most of the oscillation noise contained in the training data was oscillating in higher frequencies than most of the respiration recorded. It looked like the CNN was simply emphasizing absolute frequency more than the dynamic behavior of the frequency which was assumed to contain information about the target. The artificial expansion of the training data was changed so that each frequency contained similar amounts of both respiration and noise signals when summed over all training samples. This greatly improved the later CNNs accuracies in situations with noise oscillating close to typical respiration frequency.

The respiration-like image generated of the best CNN developed during this project clearly reflects the typical dynamic changes of respiration frequency normal to humans. This image along with a noise-like image is shown in the Selected Results chapter.

Chapter 5

Selected Results

In order to get some insight into the performance and robustness of the developed systems, this chapter will show a selection of the test results and give a brief analysis of these at the end. To make the results comparable, the data used for training, validation and testing is the same for the results shown here. The training, validation and test data is described in the first section. The accuracies achieved on the test data served as the basis for the calculation of sensitivity, specificity and ROC curves. This is because it tests the ANNs on new situations and is therefore much more interesting and realistic than accuracies achieved on the training data and the validation data which contains the same recording situations as the training data.

5.1 Training, validation and test data

5.1.1 Description

The training and validation data consisted of the following situations:

- Two Novelda employees sleeping. Recordings of some different times of night were used.
- The same employees lying on a coach and sitting in a chair, just as illustrated in figure 4.2.
- The author standing against a wall.
- Noise generated by a Lego robot oscillating at different frequencies close to respiration frequency, an oscillating roof mounted lamp and an oscillating wall mounted guitar.
- The situations above modified by artificial expansion to generate new situations with the oscillations happening at different frequencies and different rate of frequency changes.

The test data consisted of the following situations:

- Some of the situations mentioned above but from other recordings.
- Two persons sleeping next to each others.
- A third Novelda employee lying on a coach and sitting in a chair.
- Noise generated by another slightly different Lego robot also oscillating at different frequencies close to respiration frequency.
- The noise recordings were modified to cover approximately all possible respiration frequencies and to balance out the large number of respiration recordings used for testing.

5.1.2 The optimal amount

One interesting question is how much training data is needed for the best training of the developed CNN? Optimized amounts of training data means gaining good results without spending too much resources and time collecting training data. This was analyzed by running tests using only a partition of varying size of the training data, and then looking at the increased accuracy achieved due to the use of more training data. The number of training epochs was increased as the number of training samples was decreased, so that the results were comparable in the way that they had been trained for equal amounts of time. It was assumed that the accuracies had saturated within the time of each test, which by eyeballing seems to be the case. The CNN used for these tests were slightly less accurate the best achieved described later.

Figure 5.1 and 5.2 shows accuracy on training, validation and test data against training epochs. The number of samples used in test 10 is 1000 times as large as for test 1, and increases logarithmic through the 10 tests. In some cases, the test accuracy is higher than the validation accuracy, which may be a result of the heavy modified and difficult samples used for training and validation. Figure 5.3 shows the corresponding confusion matrices from the highest accuracy achieved on the test data for each of the 10 tests. The test data is used for accuracy measuring because it shows how good the CNN works in situations different from the training phase and is therefore a more realistic measure for normal use. The last epoch does not necessarily give the best result due to overfitting. Figure 5.4 shows the accuracies against the ratio of samples used to all samples. It is not clear from this plot that the accuracy saturates when all samples are used. More training data might therefore give even better results.

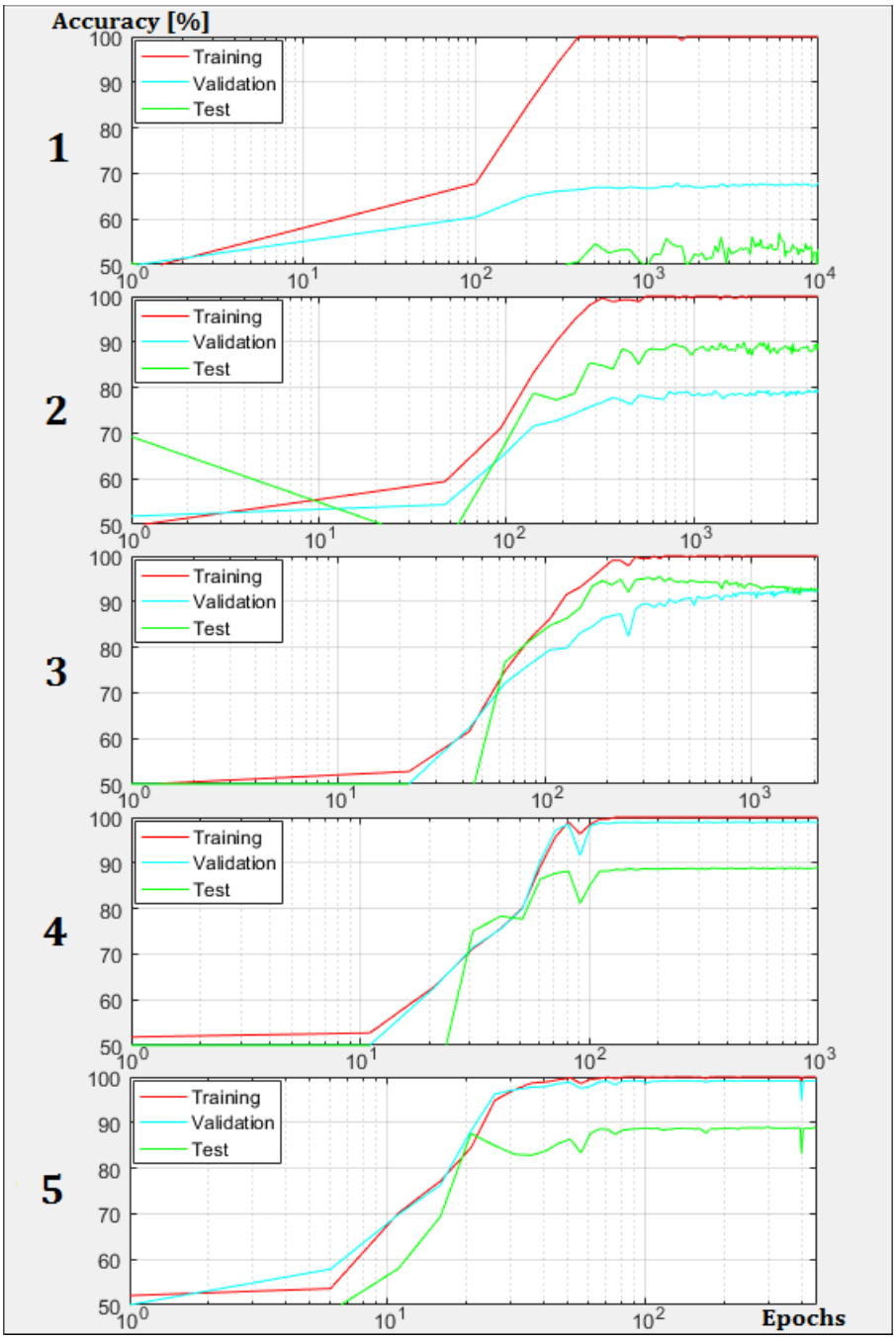

Figure 5.1: Accuracy against epochs for test 1 (1/1000 of the available training data used) to 5.

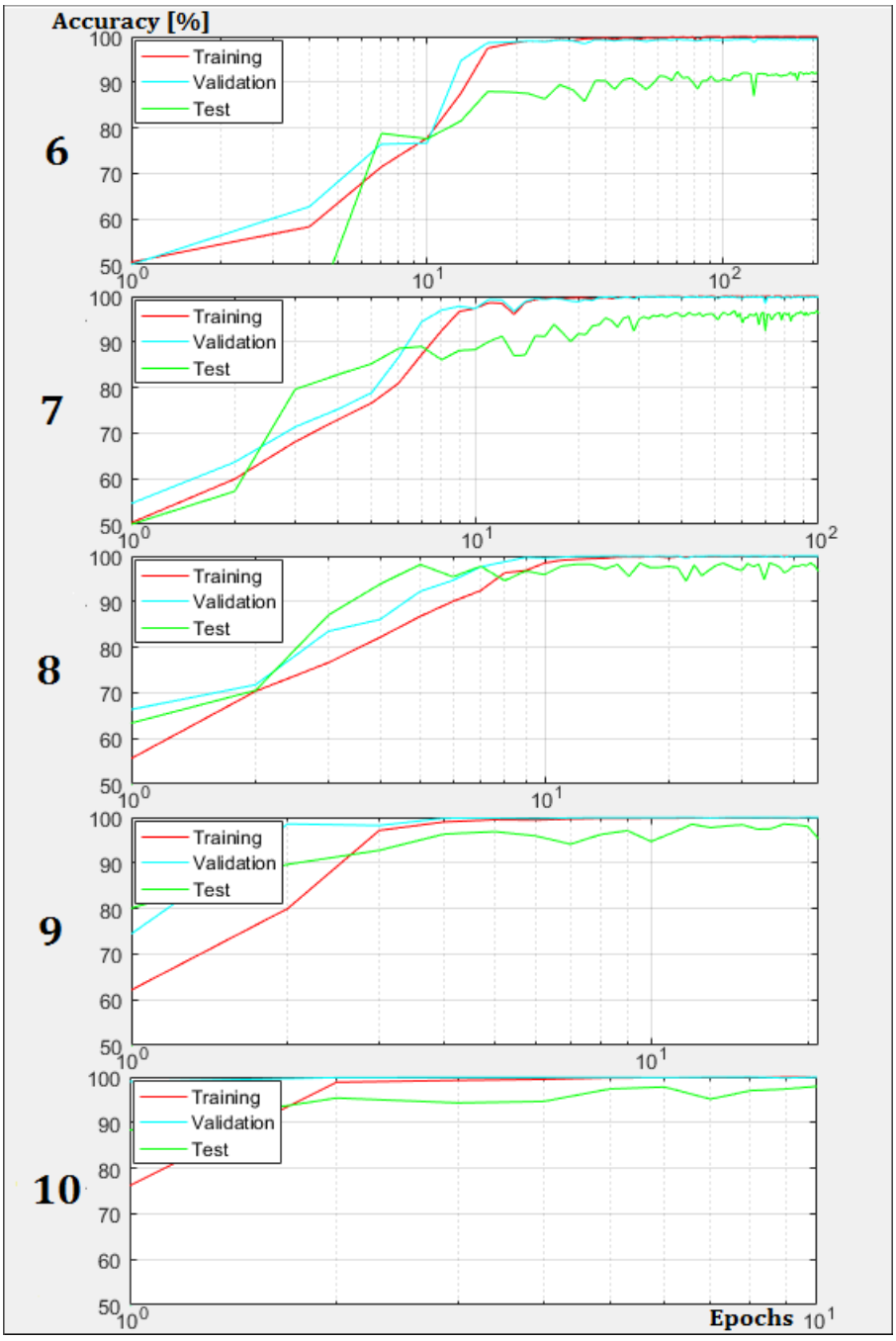

Figure 5.2: Accuracy against epochs for test 6 to 10 (All available training data used).

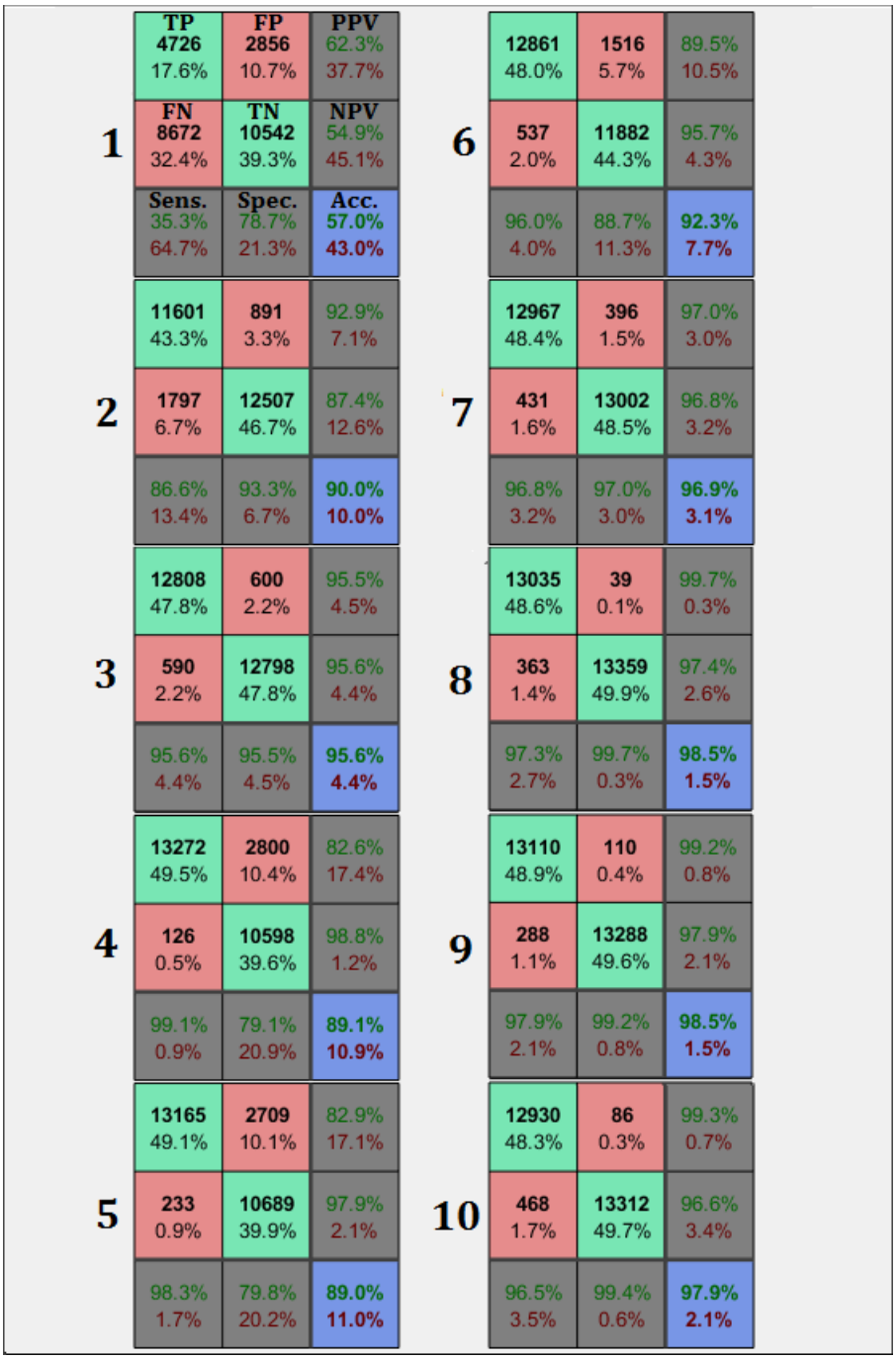

Figure 5.3: Confusion matrices for test 1 (1/1000 of the available training data used) to 10 (All available training data used).

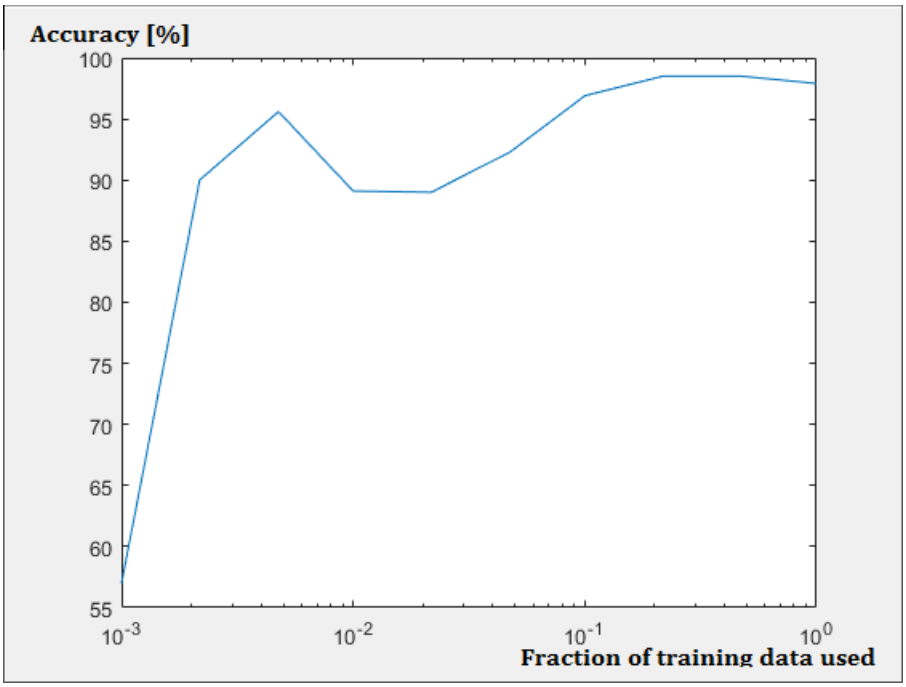

Figure 5.4: Best accuracy achieved on test data during training against fraction of the available training data used.

5.2 The convolutional networks

5.2.1 Different architectures and concepts

As described in the previous chapter, many ANN concepts were tested and its accompanying parameters tuned. Here, a selected set of results to illustrate the significance of the use of these concepts.

Convolution filter sizes

The size of the filters in the convolutional layer heavily affected the results. The first tested sizes was $5x5$ pixels, but a size of $9x9$ pixels gave much better results. The figures below illustrates the difference in a small CNN. For the first case, figure 5.5 illustrates the architecture used, 5.6 shows accuracy on training, validation and test data during training and figure 5.7 shows confusion matrix and ROC curve for the best epoch. For the second case, figure 5.8, 5.9 and figure 5.10 shows the same information. Using filter sizes of $15x15$ pixels gave slightly better results. This pattern repeated itself also for deeper CNN architectures, see the Highest accuracy achieved section for comparing with a deeper CNN with filters of $15x15$ pixels.

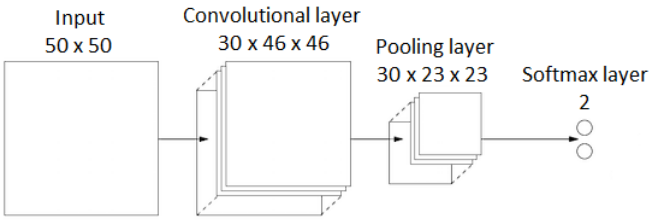

Figure 5.5: The architecture used for the 6 x 6 pixels filters tests.

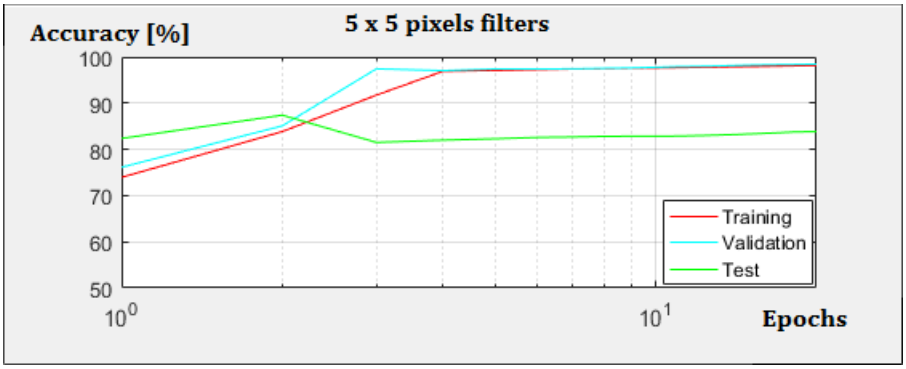

Figure 5.6: Accuracy on training, validation and test data against epochs in the 6 x 6 pixels filters test.

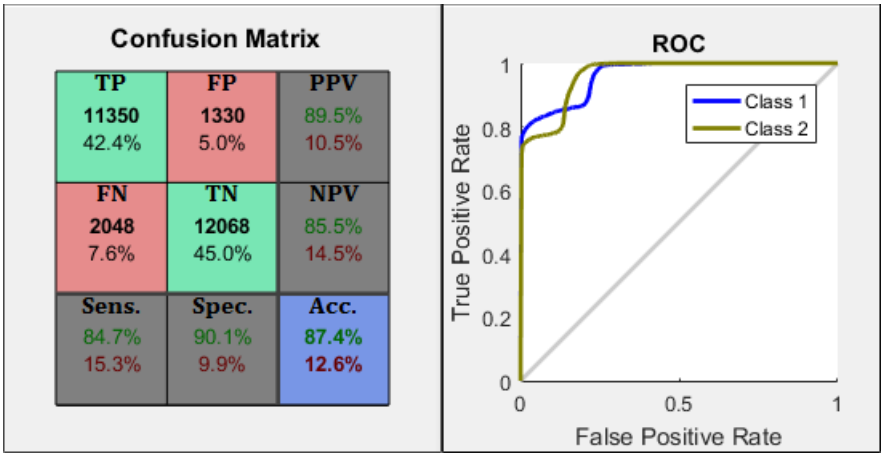

Figure 5.7: Confusion matrix and ROC plot from the epoch with the highest accuracy in the 6 x 6 pixels filters test.

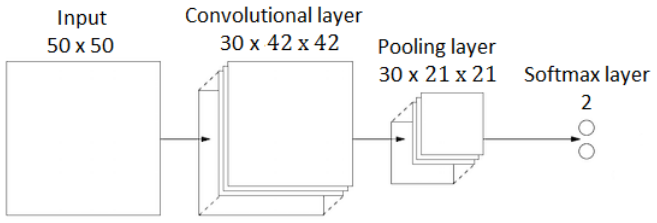

Figure 5.8: The architecture used for the 9 x 9 pixels filters tests.

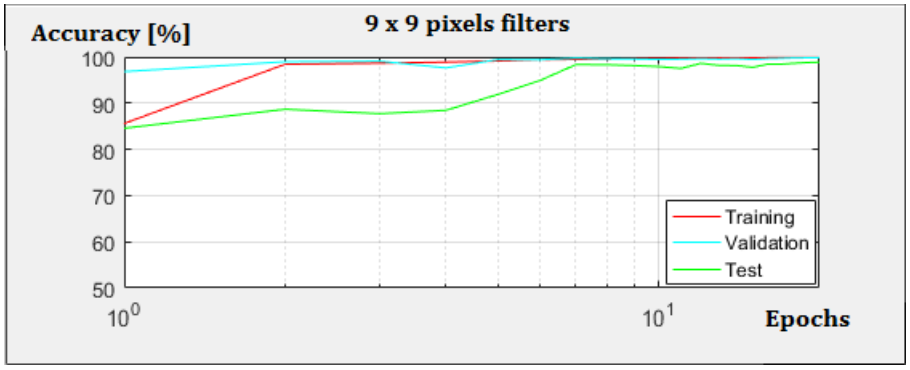

Figure 5.9: Accuracy on training, validation and test data against epochs in the 9 x 9 pixels filters test.

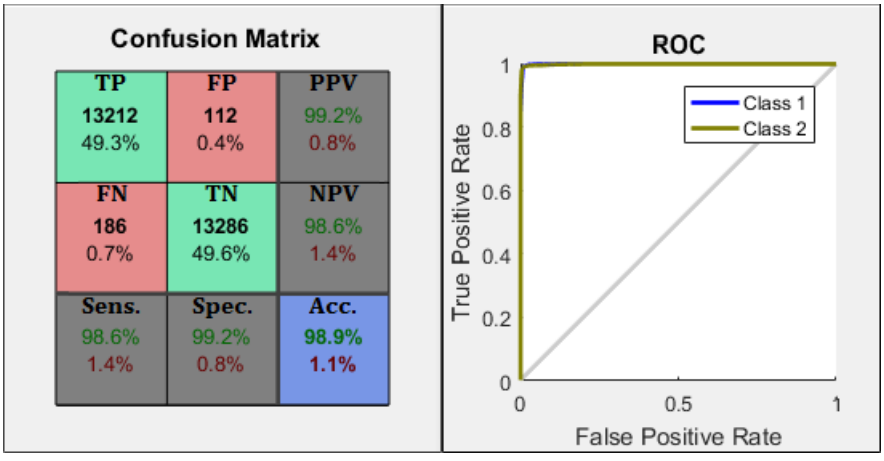

Figure 5.10: Confusion matrix and ROC plot from the epoch with the highest accuracy in the 9 x 9 pixels filters test.

Pooling layers

The CNN generating the above results included a max-pooling layer right after the convolutional layer, with pooling filters of $2x2$ pixels and the same stride length meaning no overlap. Removing the pooling layer generally gave more unstable results for most of the tests, although sometimes slightly better results. The advantage of the pooling layers down-sampling resulting in much smaller following layers, combined with the very little improvements gained by removing it, resulted in a decision of using this concept for most of the further tests. Figure 5.11 and figure 5.12 shows typical results in the same format as described and used above from training the CNN without a pooling layer.

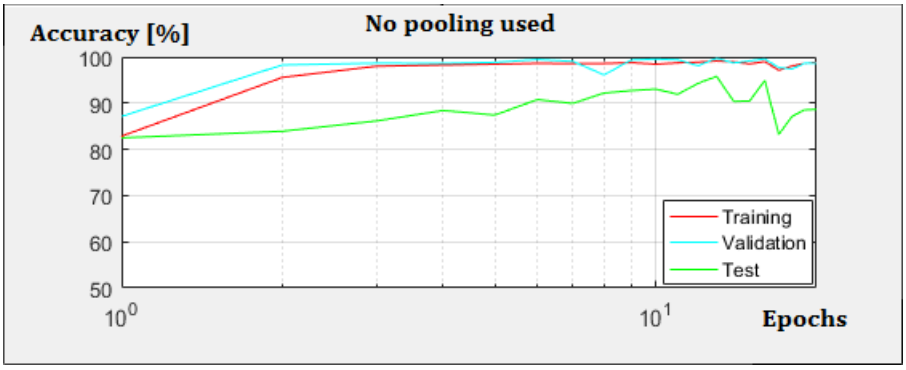

Figure 5.11: Accuracy on training, validation and test data against epochs in the no pooling test.

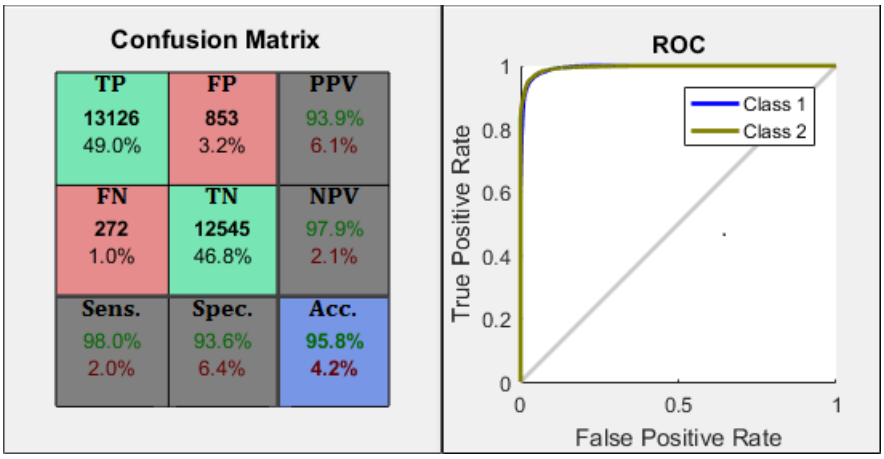

Figure 5.12: Confusion matrix and ROC plot from the epoch with the highest accuracy in the no pooling test.

Momentum

Removing momentum as an extension to the stochastic gradient descent algorithm gave as expected slower learning, illustrated in the same format as described and used above in figure 5.13 and 5.14. The sensitivity and specificity achieved would likely and seemingly increase further if training for a longer time, maybe to reach results close to results from using momentum, but from these results, clearly much slower. Again, this result is a selected typical one, after running several tests with different random initialized CNNs.

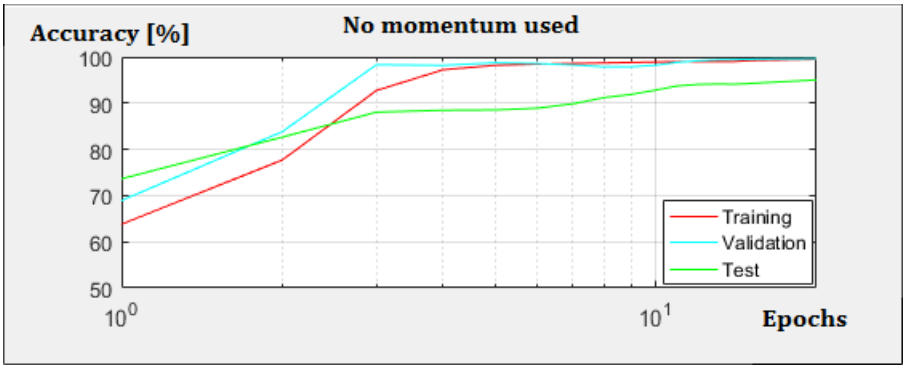

Figure 5.13: Accuracy on training, validation and test data against epochs in the no momentum test.

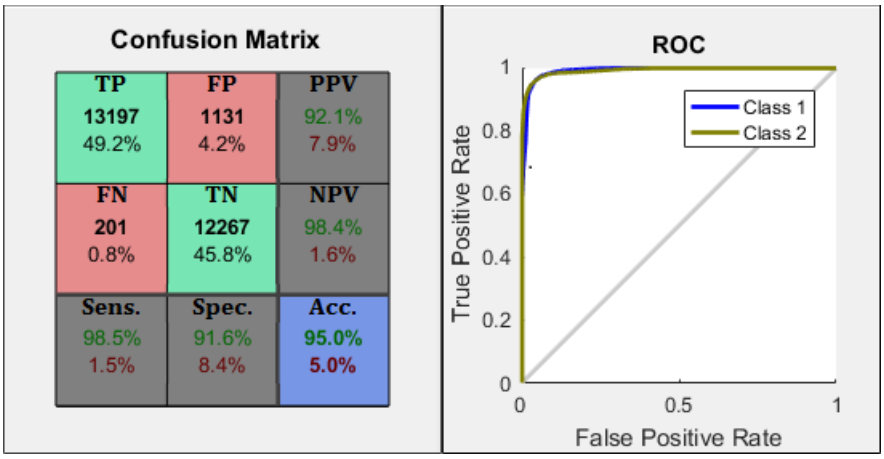

Figure 5.14: Confusion matrix and ROC plot from the epoch with the highest accuracy in the no momentum test.

5.2.2 Highest accuracy achieved

The highest accuracy achieved on the test data was with a CNN of the architecture shown in figure 5.15. This CNN was trained for 500 epochs, which took approximately 24 hours using the hardware described in the Methods and Equipment chapter.

The filters of the first convolutional layer are shown in figure 5.16 to give some insight into what shapes or features this CNN looked for, at least in its lowest level of abstraction, to achieve its results. The accuracies on the training, validation and test data are shown in figure 5.17. It is not clear from this figure that the test accuracy has saturated. Even better results might therefor be obtainable after longer time of training, but such a test would have occupied the authors computer for days and was therefore not prioritized for this project.

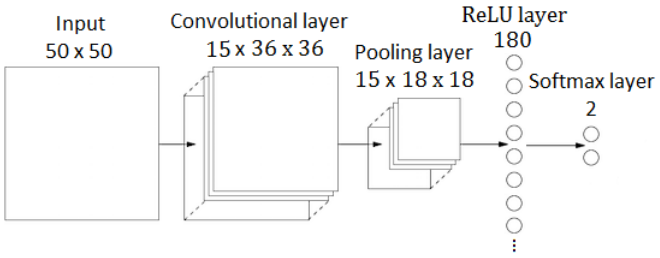

Figure 5.15: Architecture of the CNN with the highest accuracy achieved.

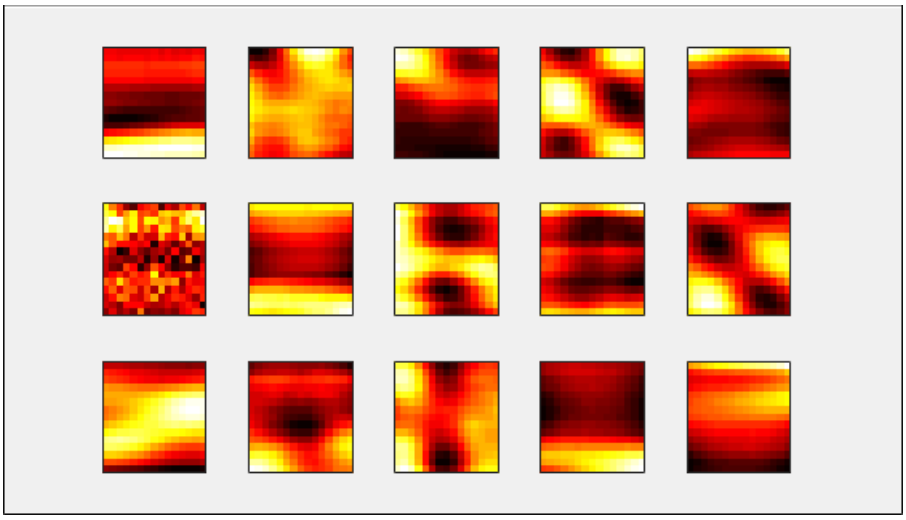

Figure 5.16: Filters of the first layer of the CNN with the highest accuracy achieved.

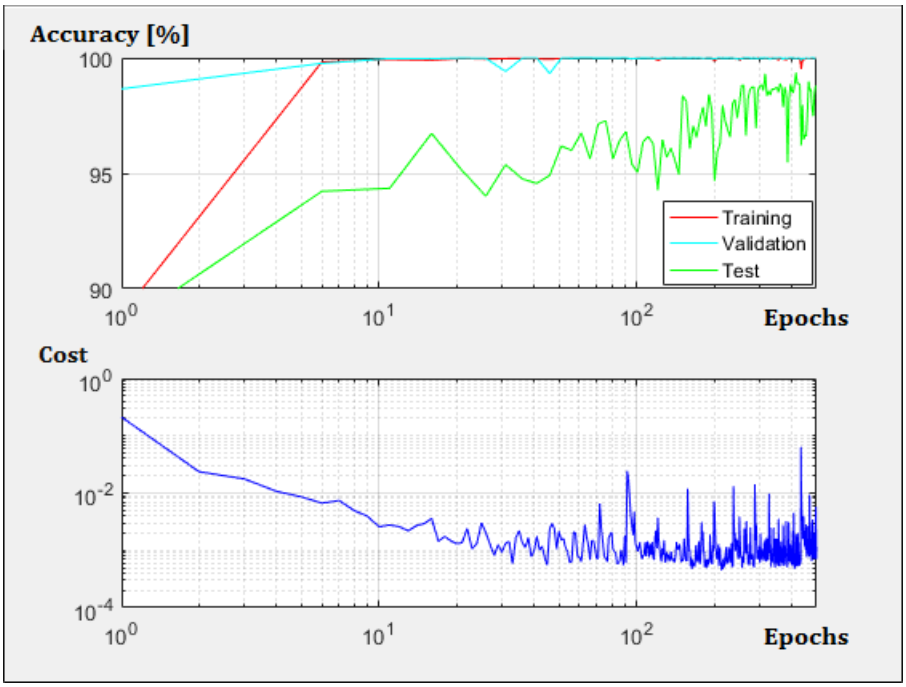

Figure 5.17: Training, validation and test accuracy during training of the CNN with the highest accuracy achieved.

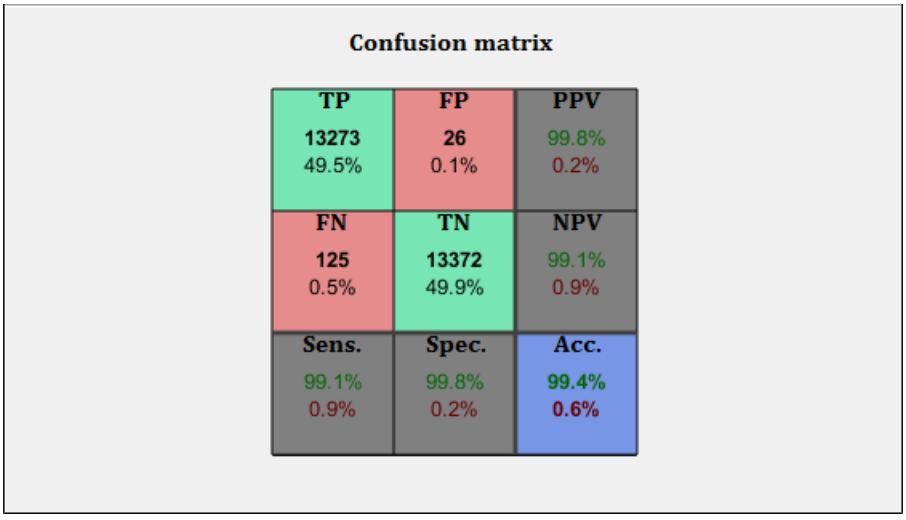

Figure 5.18: Confusion matrix of the CNN with the highest accuracy achieved.

The confusion matrix of the best classification during the testis shown in figure 5.18. The sensitivity is shown at the bottom left, 99.1%. The specificity is shown at the middle bottom tile, 99.8%. The receiver operating characteristics (ROC) are plotted on the left in figure 5.19, a section of the top left is shown on the right.

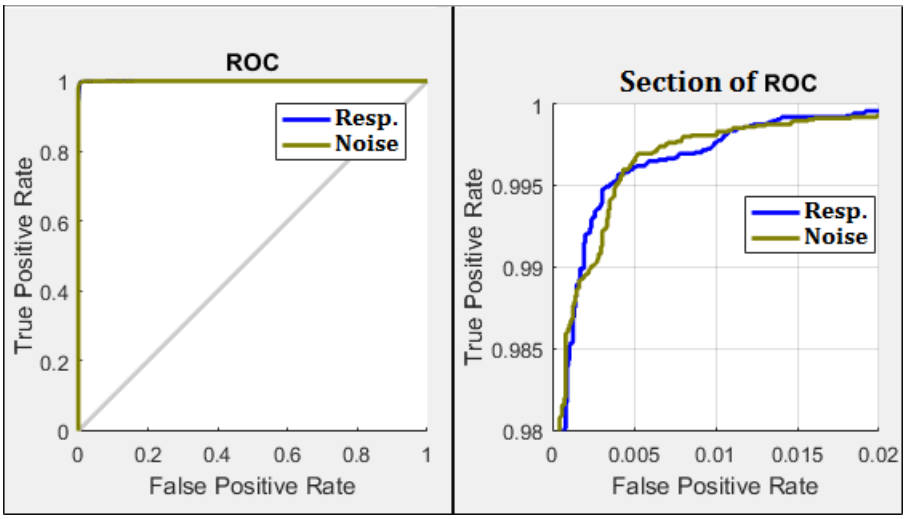

Figure 5.19: ROC plot of the CNN with the highest accuracy achieved.

5.2.3 Analysis

Most of these plots show a gradually consistent increase in accuracy for training and validation, but a more unstable and sometimes even decreasing test accuracy. This illustrates that after many epochs, the ANN overfits on the training and validation data, and gets worse at generalizing to the new data introduced in the test set. This type of plot gives a good basis for choosing the number of training epochs.

Some of these plots show situations where the accuracy on the validation set is higher than the accuracy on the training set. This may seem counter intuitive, since it is the training set that actually affects the neural connections. For the tests where a dropout layer was used, this may partially be explained by the fact that the use of such a layer decreases training accuracy since only about half of the connections to the following layer is used during training. Other reasons for this can be that the validation set is very small and therefore has a lower inner variance, or that it simply contains relatively easy samples.

5.3 Reverse engineered respiration

5.3.1 Images

This section shows the results from generating respiration-like and noise-like images using a trained CNN. In figure 5.20, generated respiration and noise samples are shown along with examples of typical samples from each case. The images are 50 x 50 pixels, the horizontal axis corresponds to time, and the vertical axis corresponds to Fourier frequency.

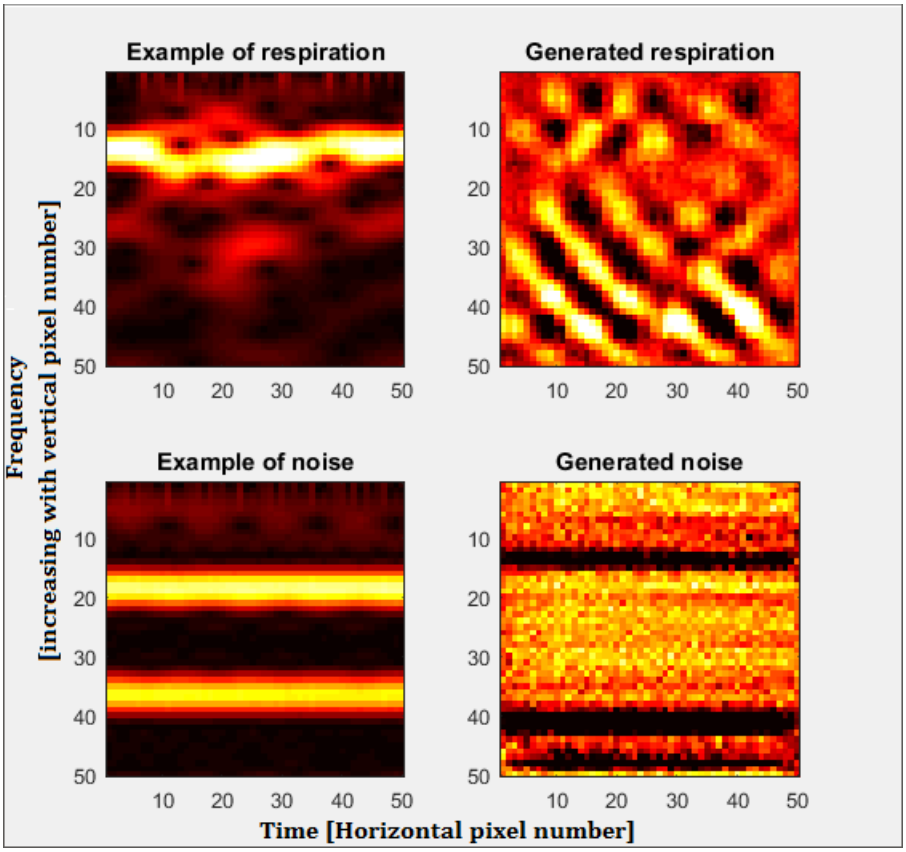

Figure 5.20: Reverse engineering: Respiration and noise signals generated from a trained CNN.

5.3.2 Analysis

These images indicate that the CNN actually looks for oscillating targets that have a time varying frequency, just like normal respiration. In the example samples, the repeated lines at double (and barely visible at triple) of the frequency of the clearest line, represent harmonic frequencies.

Chapter 6

Discussion

The objective of this project was to explore the use of artificial neural networks (ANNs) for classifying respiration from other oscillating reflectors in radar signals. This chapter gives a discussion on both the results and the work method and ends with thoughts for possible future work.

A variety of ANNs was designed, implemented and tested in Matlab. The training algorithm and the recordings used for training, validation and testing was changed several times throughout the project when new knowledge was acquired. This was a natural result of the fact that the author had no previous knowledge of ANNs. A selection of tests was therefore repeated using the latest setup and test data in order to provide comparable results for e.g. different architectures.

6.1 Results

This section will discuss these selected results and their significance.

6.1.1 Achieved sensitivity and specificity

The numbers shown in the Selected Result chapter are seemingly very good. Different testing data would however have led to different results, and even more different recording situations would presumably lead to worse looking, but even more realistic results. The sensitivity and specificity numbers given in this report is therefore not necessarily comparable to any other work.

The results will also vary each test run because of the random initializing if the ANNs. The results given are based on only a small number of repeated test runs, due to the limited time and hardware available for this project.

For the aim and scope of this project, however, the variety of recording situations was considered good enough to give a sensible comparison of the ANN

architectures studied within this project. The results also show that CNNs are clearly able to distinguish human respiration from many different objects based on the natural variety of the frequency found in human respiration, and with a high sensitivity and specificity.

6.1.2 Fully connected versus convolutional ANNs

It was early decided to allocate most of the time to the study of Convolutional Neural Networks (CNNs). The potential of fully connected ANNs was therefore explored only to a very little degree. This was a necessary prioritizing due to the time available. The decision was based on the assumption that CNNs could provide a compact and well suited architecture for the classification problem of this project, and because of the good results obtained from early tests of CNNs. Another advantage of the CNNs was that the preprocessing used was much more stable than the preprocessing for fully connected ANNs, mainly because there was no need for estimating the stationary clutter of the signals fed to the CNNs. Signals with low SNR or signals simply not containing any oscillating targets at the range analyzed gave very unstable clutter estimates, and improving the clutter estimation algorithm was considered too time consuming to be worth trying for this project.

6.1.3 Studying CNN filters

In order to acquire a deeper understanding of how the CNN actually worked, a large portion of the time available was scheduled for studying the filters in the convolutional layers. The filters clearly reflected features and shapes in the input images visible to the eye, and could therefore provide a deeper understanding of what the CNNs based the their output on.

There might still be ways that these filters are used by later hidden layers that is very hard to discover or understand, e.g. similar filters might be used to look for harmonic frequencies accompanying the most distinct frequencies. Maybe some targets generate harmonics with a higher proportion of energy than other targets, and maybe such knowledge and understanding could be used for improving the CNN further, or even give ideas to new architectures that better utilizes this information.

Studying the filters provided great help for improving the recording process and preprocessing of training samples, because it enabled recording situations that is not realistic for normal use to be discovered.

6.1.4 Reverse engineering

Reverse engineering by generating respiration-like images using a trained CNN also gave better insight into how the CNN worked. The visual patterns in the generated images was much more intuitive than the vast number of numeric

values that could be extracted from the CNN itself. As mentioned in the results chapter, the reverse engineered images indicate that the CNN actually looks for oscillating targets that have a time varying frequency, just like normal respiration. Oscillating targets with too stable frequency over time looks unnatural, and gets classified as noise. The CNN does not seem to emphasize continuous patterns as much as the fact that the frequency is simply varying. This may be a result of the relatively shallow architecture compared to many CNNs used for e.g. face recognition. The shallow architecture may not give any possibility for understanding more complicated patterns than this.

Reverse engineering also provided great help for improving the recording process and preprocessing of training samples, by the same reasons as for studying the CNN filters. Addressing the problems discovered using these methods gave huge improvements in terms of the CNNs accuracies and ability to generalize to new situations.

6.2 Methods

This section will discuss the work methods.

6.2.1 Recording data

The importance of acquiring a large and varied set of training data for testing during development was learned from the work with the project thesis [1], which also required varied recordings for use during development. For this project, Novelda employees therefore helped recording much of the data.

The recording of human respiration was mostly done by Novelda employees, but the recording of noise was only done by the author. This resulted in a much higher availability of respiration recordings than noise recordings. Because the training and test samples was balanced before use in order to give intuitive and comparable results, a very limited selection of the respiration recordings was actually used at a given time. The amount of recordings for all situations was, however, seemingly more than large enough for all tests scheduled throughout this project.

6.2.2 Effective testing

To be able to test large number of different architectures and parameters, it is beneficial to quickly obtain a minimal set of test results to be able to make decisions and move on. This happened ineffectively early in the project due to the lack of previous experience in the field, but the process was improved a lot throughout the project as experience and knowledge was gained. Early talks to the supervisors about what information is important to look for, was very helpful. The continuous improvements of the Matlab scripts used to gain

and illustrate this information, obviously also helped improving the process throughout the project.

6.2.3 Evaluation of methods

In the project thesis [1], no evaluation of methods was done before the practical and technical work was finished. The thoughts made on methods gave ideas for improvements, but only after it was too late to take advantage of them. For this project, these ideas was revisited early on, and a small amounts of time was also scheduled throughout for the evaluation of work methods, like small, very limited sprint retrospectives. This time was helpful for periodically reevaluating prioritizations and abandoning attempts that had taken too much time without leading to anything.

6.2.4 Meetings with supervisors

Because of the lack of previous experience with neural networks, knowing what to ask for was in the beginning difficult, and the supervisors were therefore not utilized as much as they could have been. This became much easier near the end of the project, when some specific results had been made and could be discussed. Possibly further work would therefore presumably be characterized by much better utilizing of supervisors from the start.

Just as with the work with [1], meetings with the Novelda supervisor through Skype and TeamViewer saved time and worked very well to review work progress and ask questions during the project.

6.3 Future work

There is a lot that could be done to follow up on the work of this project, because ANNs are a large field continuously developing, and a field not previously explored by Novelda at all. This section lists some thoughts of possible and interesting future work.

6.3.1 Artificial expansion

The artificial expansion of the training data heavily affected the results from the ANNs. This algorithm can also be designed more cleverly by using more knowledge of the signals from Novelda's radar to generate new signals that reflects the natural variations of radar signals. This makes it possible to create ANNs that can generalize better to situations not present in the training data.

For this project, the use of artificial expansion was a necessity because the frequency of most of the noise was generally higher than normal respiration frequency, resulting in problems with noise within normal respiration frequency.

The modifications done to the samples to generate respiration and noise signals with a larger variety of frequencies did not preserve the original relative distances between the harmonic components in the signal. A more realistic artificial expansion might be a good idea to explore for other use-cases, e.g. for distinguishing different kinds of respiration, possibly humans versus animals. The potential of artificial expansion for radar signals could be interesting to explore further.

6.3.2 Several convolutional layers

It would be very interesting to test CNNs with several convolutional layers connected in series because it seems logical that such CNNs might be naturally suitable for the repeating patterns in the time-frequency images. A thought of how is illustrated in figure 6.1.

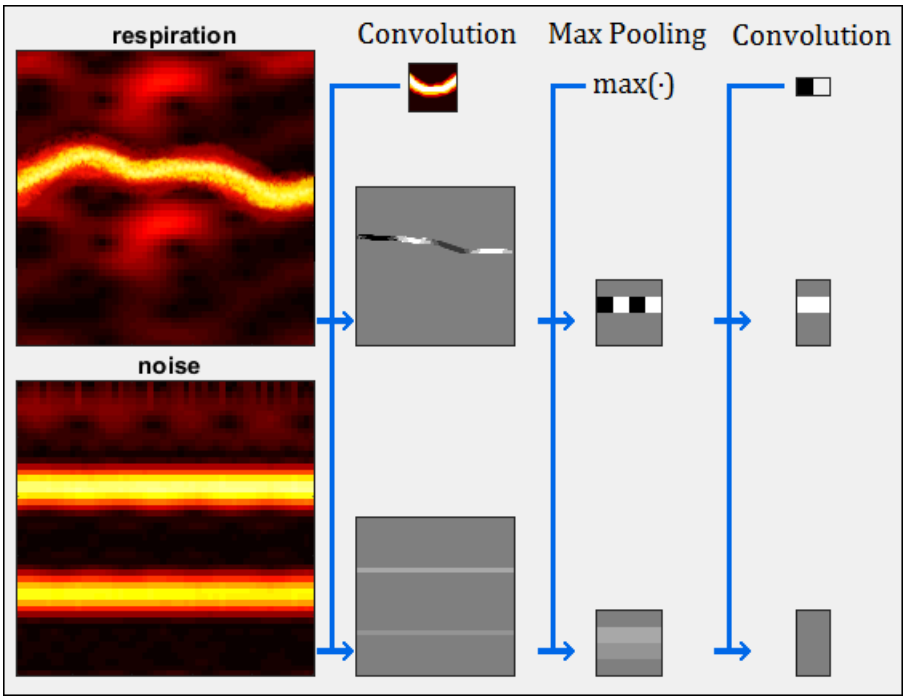

Figure 6.1: Illustration of thoughts on several convolutional layers.

A convolutional layer searches for bending lines corresponding to natural variations of human respiration. After a down-sampling by a max-pooling layer, a second convolutional layer searches for repeating patterns generated by the first convolutional layer. Now, the horizontal lines generated by an oscillating object should at the same time be suppressed, and fully connected neurons

should easily be able to distinguish the variations in frequency natural to human respiration from the even oscillations of dead object. Slightly different filters would perhaps enable the CNN to also distinguish respiration from dead objects that by some reasons also have a somewhat varying frequency. Flipping the filter of the first convolution upside down would give similar results, adding such a filter would likely be a good idea to increase performance and robustness.

Finally, it should be kept in mind that larger and more complicated architectures generally increases the problem of overfitting because the larger number of parameters contained in the system makes it easier to learn more details in samples not necessarily general to other similar samples. This is an advantage with the simple CNNs tested during this project.

6.3.3 Smarter compressing

It would be interesting to explore compressing by using pooling layers in smarter ways, e.g. the use of rectangular narrow windows to compress information that is of small or none importance for the specific classification problem, such as the mean frequency of respiration or oscillating objects. If this can be done without throwing away the important information contained in the signals, it would reduce computational complexity. Another possible way of reducing complexity can be testing modified convolutional layers that scales its filters in order to look for the same respiration patterns in different situations e.g. of high or low SNR resulting in distinct or flat ridges in the scalogram-like images used as input here. A reduction of complexity of one layer can make it possible to use inputs of a higher resolution or use more complex later layers, both of which might improve sensitivity and specificity of the classification. It can also reduce requirements for hardware or improve battery time of possible wireless devices.

6.3.4 Smarter initializing

For this project, the only approach for initializing filters was using a random Gaussian distribution, which also is the normal approach. Manually initializing at least some of the filters to reflect patterns might lead to faster training or better results because the CNN can utilize the developer's knowledge of the data in addition to learning by itself. It is a way to include our own mathematical models into the highly data-driven system in a more direct way than only deciding the CNN's architecture.

6.3.5 Achieving more stable results

To achieve stable output during normal use, some sort of tracking algorithm following the ANN to accept a small number of scattered wrongly classified samples would presumably be of interest. The development of such an algorithm was outside the scope of this project. N-version static redundancy can also

simply be achieved by letting N different or similar but random initialized ANNs vote on the result. This might also lead to better and more stable classifying.

6.3.6 Faster response time

The response time of the system if used in a real time manner was not measured or calculated because it was considered outside the scope of this project. For the preprocessing used here, it will be approximately 30 seconds because of the window length used for the Fourier transform and the number of such windows included in one input sample for the CNNs. It is possible that much lower response times can be achieved without severely compromising sensitivity and specificity of the classification. Optimizing performance in terms of response time is another interesting possible future project.

6.3.7 Learning from neural networks

Another interesting way to use artificial neural network is to train them, but instead of actually using them, rather build simple and fast classifiers based on the knowledge gained from studying and reverse engineering the neural network. This is interesting even if neural networks seems unnecessary complicated or insecure for a given task, because neural networks might be able to discover patterns and informations in data that engineers otherwise would not notice.

Chapter 7

Conclusion

This chapter concludes the discussions of results and methods made in the previous chapter. Also consult the previous chapter for thoughts on future work.

7.1 Results

The objective of this project was to explore the use of artificial neural networks for classifying respiration from other oscillating objects in radar signals. This section concludes the discussions of the results of this study.

7.1.1 Summary

Throughout the project, a variety of artificial neural networks was tested using Matlab's "Neural Networks Toolbox". The training data was recorded by the author and Novelda employees using Novelda's XeThru UWB radar. The results show that by preprocessing the radar signals to create time-vs-frequency images, and then use a convolutional neural network as classifier, human respiration can be distinguished from other oscillating objects with a high sensitivity and specificity.

7.1.2 Achieved sensitivity and specificity

The artificial neural network tested during this project that achieved the highest sensitivity and specificity, was a convolutional neural network, with a convolutional layer, a pooling layer, a hidden ReLU layer and a Softmax output layer. The architecture is illustrated in figure 5.15 in the Selected Results chapter. The sensitivity and specificity achieved on the test data was respectively 99.1% and 99.8%. The test data used is precisely described in section 5.1.1.

7.1.3 Studying CNN filters

The filters contained in the convolutional layer of the CNNs were studied to try to gain a deeper understanding of what information the neural networks actually based their classification on. The filters clearly reflected patterns also visible to the eye in the input images. The study of the filters was of great help during the project because unrealistic situations and artifacts from the recording process was often reflected in the filters. This made it possible to discover and address these problems and thereby increase the CNNs accuracy and ability to generalize to new situations.

7.1.4 Reverse engineering

A trained CNN was used to generate an input image that according to the CNN looked as much as possible like a respiration recording. This revealed that the CNN was actually looking for the variations in frequency found in normal human respiration but not in oscillating objects. Also, it only seemed to look for very local patterns, which may be a result of the relatively shallow architecture compared to the CNNs used for e.g. face recognition.

7.2 Methods

This section concludes the discussions of the work methods of this study.

7.2.1 Effectiveness and improvements

Novelda employees helped recording much of the data used in this study, resulting in a large collection of available training and testing data for the neural networks. Due to the authors lack of previous experience on the field, the testing process and the utilization of the supervisors started out inefficient but improved greatly throughout the study as experience was gained.

7.2.2 Evaluation of methods

Ideas on improvements from the authors project thesis [1] was revisited early on, and time was also scheduled throughout the study for evaluating the work and methods. This scheduled time was very helpful for periodically reevaluating prioritizations, abandoning attempts that had taken too much time without leading to anything, and evaluating the progress of the study.

References

[1] H. Blehr. *Heart rate detection with ultra wide band radar*. 2016.

[2] Lorenzo Scalise and Ingegneria Industriale. "Non Contact Heart Monitoring". In: *Advances in Electrocardiograms - Methods and Analysis* (2012), pp. 81–106. ISSN: 978-953-307-923-3.

[3] Van Nguyen et al. "Harmonic Path (HAPA) Algorithm for Non – contact Vital Signs Monitoring with IR – UWB Radar". In: *Ieee* (2013), pp. 146–149.

[4] E. M. Staderini. "UWB radars in medicine". In: *IEEE Aerospace and Electronic Systems Magazine* 17.1 (2002), pp. 13–18. ISSN: 0885-8985.

[5] URL: `https://www.xethru.com/blog/posts/xethru-respiration-sensors-pilot-project-norwegian-polic`.

[6] Chang Li. "Non-contract Estimation of Respiration and Heartbeat Rate using Ultra-Wideband Signals". In: *Thesis* (2008).

[7] J. C. Lin. "Noninvasive microwave measurement of respiration". In: *Proceedings of the IEEE* 63.10 (Oct. 1975), pp. 1530–1530. ISSN: 0018-9219.

[8] J. C. Lin, E. Dave, and J. Majcherek. "A noninvasive microwave apnea detector". In: *Proceedings of the San Diego Biomedical Symposium* (1977), 441–443.

[9] J. C. Lin et al. "Microwave Apexcardiography". In: *IEEE Transactions on Microwave Theory and Techniques* 27.6 (June 1979), pp. 618–620. ISSN: 0018-9480.

[10] K. H. Chan and J. C. Lin. "Microprocessor-based cardiopulmonary rate monitor". In: *Medical and Biological Engineering and Computing* 25.1 (1987), pp. 41–44. ISSN: 1741-0444.

[11] M. Nowogrodzki, D. D. Mawhinney, and H. F. Milgazo. "Non-invasive microwave instruments for the measurement of respiration and heart rates." ENGLISH. In: *Proc IEEE Natl Aerosp Electron Conf* 1984.2 (1984), pp. 958–960.

[12] J. Seals, S. R. Crowgey, and S. M. Sharpe. *Electromagnetic Vital Signs Monitor, Tech. Rep. Final Report Project A-3529-060*. Atlanta: Georgia Tech Research Institute Biomedical Division, 1986.

[13] Kun-Mu Chen et al. “Microwave life-detection systems for searching human subjects under earthquake rubble or behind barrier”. In: *IEEE Transactions on Biomedical Engineering* 47.1 (Jan. 2000), pp. 105–114. ISSN: 0018-9294.

[14] E. F. Greneker. “Radar sensing of heartbeat and respiration at a distance with applications of the technology”. In: *Radar 97 (Conf. Publ. No. 449)*. Oct. 1997, pp. 150–154.

[15] G. Ossberger et al. “Non-invasive respiratory movement detection and monitoring of hidden humans using ultra wideband pulse radar”. In: *Ultra Wideband Systems, 2004. Joint with Conference on Ultrawideband Systems and Technologies. Joint UWBST IWUWBS. 2004 International Workshop on*. May 2004, pp. 395–399.

[16] M. Paksuniemi et al. “Wireless sensor and data transmission needs and technologies for patient monitoring in the operating room and intensive care unit”. In: *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*. Jan. 2005, pp. 5182–5185.

[17] S. Venkatesh et al. “Implementation and analysis of respiration-rate estimation using impulse-based UWB”. In: *MILCOM 2005 - 2005 IEEE Military Communications Conference*. Oct. 2005, 3314–3320 Vol. 5.

[18] K. Kang, I. Maravic, and M. Ghavami. “Ultra-Wideband Respiratory Monitoring System”. In: *Ultra Wideband Systems, Technologies and Applications, 2006. The Institution of Engineering and Technology Seminar on*. Apr. 2006, pp. 191–195.

[19] Rivera et al. “Feasibility of dual UWB heart rate sensing and communications under fcc power restrictions”. In: *Proceedings of the 14th European Signal Processing Conference (EUSIPCO 2006)*. Florence, Italy, Sept. 2006.

[20] S. Gezici. “Theoretical Limits for Estimation of Periodic Movements in Pulse-Based UWB Systems”. In: *IEEE Journal of Selected Topics in Signal Processing* 1.3 (Oct. 2007), pp. 405–417. ISSN: 1932-4553.

[21] C. G. Bilich. “Feasibility of dual UWB heart rate sensing and communications under fcc power restrictions”. In: *Proceedings UWB Radio Technology Workshop/IET 2007 Symposium on UWB*. 2007.

[22] H. Langen. *Ultra-Wideband Radar Simulator for classifying Humans and Animals based on Micro-Doppler Signatures*. 2016.

[23] Michael A. Nielsen. “Neural Networks and Deep Learning”. In: Determination Press, 2015.

[24] S.I. Fox. *Human Physiology*. McGraw-Hill international edition. McGraw-Hill, 2013. ISBN: 9781259060540.

[25] *Photobucket.com*. URL: `http://i1304.photobucket.com/albums/s523/sarahlynn244/Anatomy%20142/AlveolitheRespiratoryMembrane2_zps9b813ff5png`.

[26] Øyvind Aardal et al. “Chest movement estimation from radar modulation caused by heartbeats”. In: *2011 IEEE Biomedical Circuits and Systems Conference, BioCAS 2011* 4 (2011), pp. 452–455.

[27] URL: `https://www.xethru.com/`.

[28] P. Y. Simard, D. Steinkraus, and J. C. Platt. “Best practices for convolutional neural networks applied to visual document analysis”. In: (Aug. 2003), pp. 958–963.

Appendices

A Selected MATLAB code

This appendix shows selected parts of the Matlab code written during the project. It is estimated that 2 000 lines of code was written.

A.1 Generating reverse engineered respiration sample

```
% Script to generate the most possible respiration-like image.
clc

% Parameters
image_resolution_x  = 50;
image_resolution_y  = 50;
step_size_factor    = 1e-4;

% Inputs
data = load('CNN_3');
trainedNet = data.net.net;

% Run loop
imageres    = zeros(image_resolution_x,image_resolution_y,1);
imagenoise  = zeros(image_resolution_x,image_resolution_y,1);
fres        = -Inf;
fnoise      = -Inf;
while 1
    % Generate gaussian white noise
    image_addOn = step_size_factor*randn(size(image1res));

    % Update respiration image
    features = activations(trainedNet,imageres+image_addOn,8);
    if features(1)-features(2)>fres
        imageres = imageres+image_addOn;
        imageres = imageres-mean(imageres);
        fres = features(1)-features(2);
    end

    % Update noise image
    features = activations(trainedNet,imagenoise+image_addOn,8);
    if features(2)-features(1)>fnoise
        imagenoise = imagenoise+image_addOn;
        imagenoise = imagenoise-mean(imagenoise);
        fnoise = features(2)-features(1);
    end
end
```

A.2 Create time-vs-frequency images

Create time-vs-frequency images of energy content of signals in the IQ domain.

```
counter = 1;
for k=1+floor(WindowLength_s*Fs/2):floor(JumpDistance_s*Fs):input_signal_length-floor(WindowLength_s*Fs/2)
    x = input_signal(k-floor(WindowLength_s*Fs/2):k+floor(WindowLength_s*Fs/2)-1);
    x = x-mean(x);
    x = x/std(x);
    L = length(x);
    fft_win = hann(L)';
    fft_RPM = Fs*(0:(fft_oversampling_rate*L/2))...
            / L*60/fft_oversampling_rate;

    % FFT
    input_signal_dft_2sided = ...
        abs(fft(fft_win.*x,fft_oversampling_rate*L)/L);
    input_signal_dft_1sided = ...
        input_signal_dft_2sided(1:fft_oversampling_rate*L/2+1);
    input_signal_dft_1sided = abs(input_signal_dft_1sided(1:150));
    fft_RPM                 = fft_RPM(1:150);

    % Create time-vs-frequency images of energy content
    figure(200);plot(fft_RPM,input_signal_dft_1sided);
    Image(:,counter) = imresize(input_signal_dft_1sided,[1 image_resolution_y])';
    if counter>=image_resolution_x
        x_temp(1+counter-image_resolution_x,:,:) = ...
            Image(:,1+counter-image_resolution_x:counter);
    end
    counter = counter + 1;
end
I=find(~squeeze(x_temp(:,1,1)));
x_temp(I,:,:) = [];
```

A.3 Artificial expansion example

This code shows an example of artificial expansion for shifting the images vertically (along the frequency axis).

```
function [X_train_exp,T_train_exp] =
    dataProc_artificialExpansion2D_3(X_train,T_train)
% Artificial expansion of scalogram-like 2D images of
    radar signals.
expFactor   = 21;
X_train_exp = zeros(size(X_train,1),size(X_train,2),...
                    size(X_train,3),size(X_train,4)*
                       expFactor);
T_train_exp = zeros(size(T_train)*expFactor);

fprintf('Art Exp sample:')
for k=1:size(X_train,4)
    x_train = X_train(:,:,:,k);
    t_train = T_train(k);

    % Periodically print progress
    if mod(k,100)==0
        fprintf('%d ',k)
    end

    % Original
    X_train_exp(:,:,:,expFactor*(k-1)+1) = x_train;
    T_train_exp(      expFactor*(k-1)+1) = t_train;

    % Shifted
    for m=1:20
        X_train_exp(:,:,:,expFactor*(k-1)+m+1) = [
           x_train(2*m:end,:);x_train(1:2*m-1,:)];
        T_train_exp(      expFactor*(k-1)+m+1) =
           t_train;
    end
end
fprintf('\n')
end
```

A.4 Using the trainNetwork function

This code illustrates an example uf using the built in trainNetwork Matlab function from the "Neural Network Toolbox".

```
layers = [...
    imageInputLayer([50 50 1],'Name','Input');
    convolution2dLayer(9,15,'Name','Conv1');
    reluLayer('Name','ReLU1');
    maxPooling2dLayer([2 2],'Name','Pool1');
    fullyConnectedLayer(180,'Name','Forw1');
    reluLayer('Name','ReLU2');
    dropoutLayer('Name','Drop1');
    fullyConnectedLayer(2,'Name','Forw2');
    softmaxLayer('Name','Softmax');
    classificationLayer('Name','Output')];
layers(2,1).WeightL2Factor = 0;

options = trainingOptions('sgdm',...
    'MaxEpochs',200,...
    'InitialLearnRate',0.02,...
    'LearnRateSchedule','none',...
    'LearnRateDropFactor',0.1,...
    'LearnRateDropPeriod',30,...
    'Momentum',0.9,...
    'L2Regularization',0.0001,...
    'MiniBatchSize',128,...
    'CheckpointPath','C:\Users\Harald\',...
    'OutputFcn',@customOutputFcn);

[trainedNet,traininfo] = ...
    trainNetwork(X_train,T_train,layers,options);
```