NTNU
Norwegian University of
Science and Technology

ANN based classification of humans and animals using UWB radar

Børge Wiik

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Tor Engebret Onshus, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Preface

This Master thesis is carried out at NTNU in cooperation with Novelda AS during the spring semester of 2018. The main objective of the thesis is to classify radar data using a neural network. It continues the work of an earlier project written by the author in the fall of 2017 [11], where a small Convolutional Neural Network (CNN) was tested on pulse doppler radar data captured by Noveldas X4 radar chip. Novelda provided the opportunity to capture radar data at a film studio using the X4 radar, as well as a Matlab framework for visualizing and extracting features from the raw radar data. The project work succeded in extracting features with the given framework. Code was written that formatted these features into input images to a neural network and code for a neural network architecture was written and succesfully tested upon.

In this thesis, the code from the project work has been improved and made more efficient. A smaller and less computationally expensive framework has been written to include only parts of the Novelda framework that are necessary for extracting doppler data. The knowledge and experience obtained during the project work has been used to come up with new ideas to improve classification of the radar data. In parallel with the work of this thesis, a simulator that generates radar data was developed as part of a master thesis by Anders Liland at the Electronics Systems Design and Innovation department of NTNU [7], also in cooperation with Novelda AS. It was placed at disposal in order to generate more data. The development and arranging of input images from radar data to the neural network was a time consuming effort and played a major role in this work. Different neural network architectures and image designs were tested to gain deeper knowledge of the problem at hand with the final goal of predicting and classifying radar images with high certainty.

Acknowledgment

I would like to thank Novelda AS and Jan Roar Pleym in particular for providing me the opportunity to work with deep learning in a very interesting context. His sharing of knowledge within the field of radar technology has been most helpful in the process of understanding the radar module used in this work. I would also like to thank my supervisor at NTNU, Tor Onshus, for helping me structure my working process and the Department of Engineering Cybernetics at NTNU for providing GPU resources.

Summary and Conclusion

Micro-Doppler signatures take advantage of Doppler information in radar data to create time VS. frequency information. Such signatures are interpretable by the human eye with visible features such as frequencies of swinging arms and legs as well as radial velocities of moving objects. This work explores the possibilities of distinguishing human and animal signatures by using micro-Doppler data as input to a neural network classifier, more specifically a CNN.

A simulator was used to generate radar data of humans and dogs. This data is noise-free and formed the basis for training and testing the neural network. Furthermore, a second test set was developed based on real radar data. The real data was created from recordings of a human and a dog in various situations using Noveldas X4 radar. This was done in a controlled test environment to minimize noise and disturbances on the data. Micro-Doppler signatures are retrieved from the radar data by a feature extraction process using a series of digital signal processing steps. These are appropriately formatted to serve as input data to a CNN. Two types of images with different resolutions are studied. Large images of size 256×256 contain high resolution frequency content, whereas the downsampled images of size 32×32 contain contours of the same content with poor resolution. Machine learning methods such as L2-regularization and dropout are explored in efforts to increase testing performance of the neural network.

The neural network was found to train more easily when using images of micro-Doppler signatures over several seconds. These images contain frequency information such as multiple cycles of moving limbs that can be picked up by the network to distinguish a human from a dog. The alternative of using images with only small variations in frequency content proved to be hard to classify as no general features that can be extracted are visible in the images.

Testing the images on the different CNN configurations revealed that the human images were typically more easily correctly classified than the dog images. The greater access to diverse human scenario radar data than dog data makes it difficult for the network to learn

general dog features. The best network predictions were made on the 32×32 images. 100% classification accuracy was achieved on the simulated test images when using the dropout technique as well as when using the dropout technique together with L2-regularization. These cases showed strong levels of predictions on both human images and dog images. This suggests that some general features that could separate the two classes were learned by the CNN during training.

The real data was poorly classified by the network configurations tested upon. This is as expected as the training of the CNN was based on simulated images only. The noise in the real images is non-existent in the simulated images. Relating real and simulated images is therefore difficult.

As a conclusion, simulated micro-Doppler signatures over several seconds can be classified with high certainty. A high resolution is not necessary to capture important features in the data. Due to a greater access to diverse radar data on humans, general features in human images are more easily learned by a neural network than what is the case with dog images. Real data can not be classified with the methods used in this thesis. For this to be realistic, more resemblance between real and simulated data is necessary or real images must be included when training the neural network, which requires a greater access to real data.

Oppsummering og konklusjon

Micro-Dopplersignaturer utnytter Dopplereffekten i radardata til å skape tid VS. frekvensinformasjon. Slike signaturerer kan tolkes av menneskeøyet med synlige detaljer som svingefrekvenser på armer og ben, samt radiale hastigheter, hos bevegelige objekter. Dette arbeidet utforsker mulighetene til å separere menneske- og dyresignaturer ved å bruke micro-Dopplerdata som inndata til et nevralt nett, nærmere bestemt et CNN.

En simulator ble brukt til å generere radardata av menneske og hund. Disse dataene er uten støy og ble brukt som utgangspunkt til trening og testing av det nevrale nettet. I tillegg ble enda et testsett dannet basert på ekte radardata. De ekte dataene ble skapt fra opptak av menneske og hund i forskjellige situasjoner ved bruk av Noveldas X4-radar. Dette ble gjort i et kontrollert testmiljø for å begrense støy og forstyrrelser på dataene. Micro-Dopplersignaturer er hentet ut av radardatenene gjennom en signalbehandlingsprosess. De er videre formattert til å kunne brukes som inndata til det nevrale nettet. To typer bilder med ulike oppløsninger er utforsket. Store bilder av størrelse 256×256 inneholder frekenvsinnhold av høy oppløsning, mens nedsamplede bilder av størrelse 32×32 inneholder konturer av det samme innholdet, men med lavere oppløsning. Maskinglæringsmetoder som "L2-regularisering" og "dropout" er utforsket i forsøk på å forbedre klassifiseringsevnen til det nevrale nettet.

Det nevrale nettet trente enklere ved bruk av micro-Dopplersignaturer over flere sekunder. Disse bildene inneholder frekvensinformasjon som representerer flere sykluser med svingende armer og ben, som videre kan plukkes opp og brukes av det nevrale nettet til å skille mellom menneske og hund. Et alternativ er å bruke bilder med små frekvensvariasjoner, men slike bilder viste seg å være vanskelig å klassifisere ettersom ingen generelle detaljer i bildene er synlige som et nevralt nett kan lære seg.

Testing av bildene på de ulike CNN-konfigurasjonene avslørte at menneskebilder ofte var enklere å klassifisere riktig enn hundebilder. Større tilgjengelighet av varierte menneskedata enn hva gjelder hundedata gjør det vanskeligere for det nevrale nettet å lære generelle hun-

deegenskaper. De beste nettverksprediksjonene ble gjort på bildene av størrelse 32×32. 100% klassifiseringsnøyaktighet ble gjort på det simulerte testsettet når dropout-teknikken ble brukt under treningen av det nevrale nettet samt når dropout ble brukt sammen med L2-regularisering. Disse tilfellene viste sterke klassifiseringsevner på både menneske- og hundebildene. Dette tyder på at det nevrale nettet lærte noen generelle detaljer i bildene gode nok til å kunne separere de to klassene.

De ekte dataene ble klassifisert dårlig av de nettverkskonfigurasjonene som ble testet på i dette arbeidet. Dette er som forventet ettersom treningen av nettet var basert kun på simulerte data. Støy i de ekte dataene er ikke-eksisterende i de simulerte dataene, noe som gjør de vanskelige å relatere til hverandre.

Det konkluderes med at simulerte micro-Dopplersignaturer over flere sekunder kan klassifiseres med høy nøyaktighet. Høy oppløsning er ikke nødvendig for å fange viktige egenskaper i dataene. Grunnet større tilgang til varierte radardata på mennesker er det enklere for et nevralt nett å lære seg generaliserte detaljer i menneskebilder enn hva gjelder hundebilder. Ekte data kan ikke klassifiseres med de metodene brukt i dette arbeidet. Dette kan gjøres realistisk ved å gjøre simulerte og ekte data mer like, eller ved å inkludere ekte data i treningen av det nevrale nettet, noe som krever større tilgang til ekte data.

Acronyms

ADC Analog-to-Digital Converter.

BGD Batch Gradient Descent.

CNN Convolutional Neural Network.

EM electromagnetic.

IID Independent and Identically Distributed.

LoS Line of Sight.

PRF Pulse Repetition Frequency.

PRI Pulse Repetition Interval.

ReLU Rectified Linear Unit.

RF radiofrequency.

SGD Stochastic Gradient Descent.

SNR Signal to Noise Ratio.

STFT short-time Fourier transform.

UWB Ultra Wide Band.

Contents

Chapter 1

Introduction

1.1 Motivation

The technology behind Novelda's X4 radar performs far beyond traditional radar systems when it comes to compact size, power consumption, accuracy, speed and low emissions [1]. The single-chip technology has the ability to detect presence, i.e. of a human breathing. This can be used in applications such as light switching or heart rate monitoring in hospitals. It can measure distances and detect motions through objects. This can be used in safety and security solutions such as alarm systems. If the radar can distinguish between humans and animals, the alarm can be set off only in the presence of humans, allowing pets to walk around freely.

A radar can detect moving objects by taking advantage of the Doppler effect. The velocities of the moving objects can be distinctly represented in the frequency domain, where typical signatures can be extracted and used to classify the target of interest. Neural networks are potentially very powerful classifiers that, with proper data, can learn to extract relevant features on their own. This makes neural networks an interesting study to further investigate as classification method in relation with Doppler data.

1.2 Problem Despcription

Novelda Ultra Wide Band (UWB) radar is capable of detecting humans and animals by taking use of the Doppler effect. A master thesis was written in the spring of 2016 on this subject [6] and in the fall of 2017 a CNN was implemented to classify the radar images as part of a project work [11].

In the master thesis of spring 2016, testing was performed on a radar (X2) of limited Doppler and range capabilities. A new radar (X4) without these limitations was tested in the project work of fall 2017. Limitations in the X4 recordings were partly low Signal to Noise Ratio (SNR), challenges with range estimation to target and few measurements available.

In this master thesis we want to develop the CNN algorithm further without the limitations experienced in the project work. This requires a new data foundation and can be solved by recording new measurements and by using a simulator which is developed in parallel with this thesis.

Tasks:

- Record new measurements on humans and animals using the X4 radar.
- Use a simulator to generate synthetic radar data.
- Further develop the CNN algorithm from fall 2017.
- Train and test the CNN classifier on micro-Doppler signatures extracted from radar data acquired.

1.3 Earlier work

Novelda AS has been involved in multiple thesises concerning analysis and classification of radar data. The older X2-radar has been studied in [12], where micro-Doppler signatures of humans and dogs were extracted and compared. In [6], synthetic data created from a mathematical model of a radar and point scatterer models of humans and animals was used in order to train a fully connected neural network. It was tested on real data acquired from the X2 radar. In [2], a CNN was used to classify respiration of humans and animals, detected by the X2 radar. Similar studies were conducted in [11], only with micro-Doppler signatures of walking humans and animals extracted from Novelda X4 radar.

1.4 Gait and bulk motion

Two terms that are frequently used throughout this thesis are *gait* and *bulk motion*. Gait is by *www.dictionary.com* defined as *"a manner of walking, stepping or running"* and will be used in this work to describe locomotion through the movement of swinging limbs of humans and animals. A gait cycle corresponds to one period of swinging arms or legs. Similarly, gait

frequency is the frequency at which arms or legs swing. The bulk motion is the motion of the main part of the body, which in this work corresponds to the human and dog torso.

1.5 Outline

The rest of the thesis is structured as follows. Chapter 2 gives an overview of relevant pulse Doppler radar theory as well as the feature extraction method used to extract Doppler information from the radar data. Chapter 3 covers the neural network theory necessary to grasp the groundwork forming the basis for the tests carried out in this thesis. Chapter 4 includes methods regarding data acquisition, input data design and design of the CNN. Chapter 5 presents the testing and results of the neural network with the input images created. The work of this thesis is further discussed in chapter 6, where thoughts on future work also are presented...

Chapter 2

Pulse Doppler Radar and Feature Extraction

This chapter provides basic theory needed to understand the pulse Doppler radar used in this thesis. It also covers a feature extraction process to acquire micro-Doppler signatures from radar data. The theory is based on Richards, Scheer and Holm's book *Principles of Modern Radar* [9], which provides a comprehensive introduction to modern radar systems. The feature extraction process follows the procedure of work done in [6].

All symbols used when presenting the radar theory are listed in table (2.1). Similarly are the symbols used as part of the feature extraction process listed in table (2.2).

2.1 Radar principles

Radar, short for *Radio Detection And Ranging*, is a system that uses electromagnetic (EM) waves in the radiofrequency (RF) region to detect objects of interest. Early radar systems were limited to detecting targets and measuring target range. The field of radar systems have evolved into modern radars, capable of tracking, imaging and classification of targets.

2.1.1 Radar Subsystems

A radar must be comprised of major subsystems such as a transmitter, an antenna and a receiver, as depicted in figure (2.1). It shows an EM wave being generated by the transmitter and radiated by the antenna. The transmitted signal reaches the target and is reradiated back into the environment. Some of the reradiated signal is captured by the antenna. The target echo will not be the only signal intercepted by the antenna, as all other surfaces also cause echoes. Such unwanted signals are called *clutter*. The captured signal is processed in

the receiver, where it is amplified and further converted to a digital signal by an Analog-to-Digital Converter (ADC). The transmitter and receiver are isolated by a switch to protect the receiver from high-power waves generated in the transmitter. It also provides a connection point so that the antenna can be used on a shared-time basis by the transmitter and receiver.

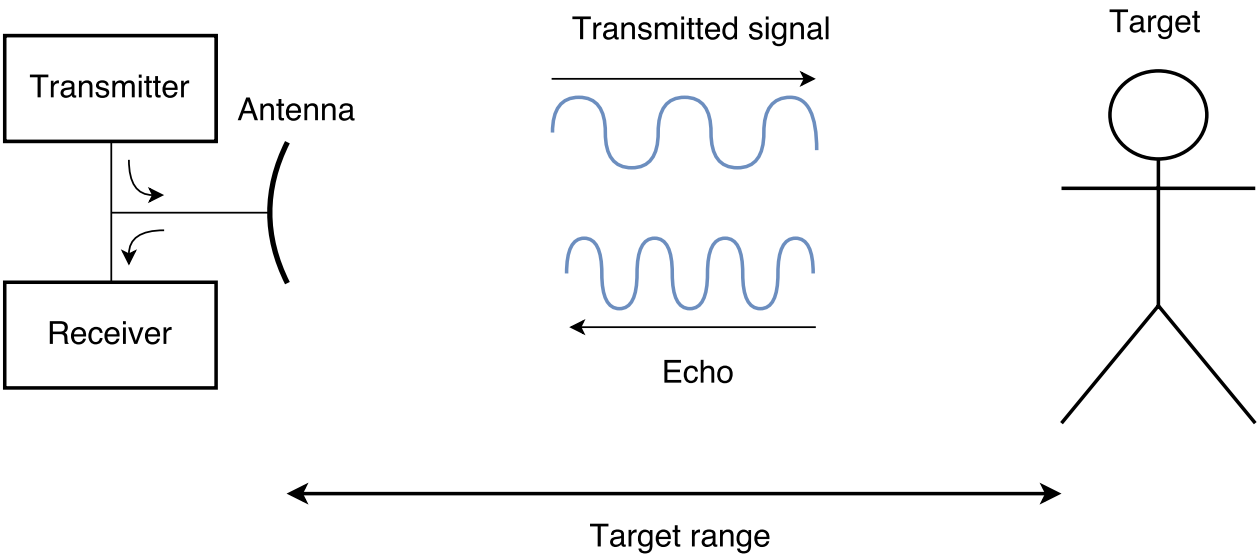


Figure 2.1: Radar principle [11]

In a pulse Doppler radar, the transmitter and receiver work on a shared-time basis. EM waves are transmitted for a short time duration where only the transmitter is connected to the antenna. This is followed by a listening period where the receiver is connected and reradiated signals are allowed in, before another pulse is transmitted. The time duration of one such pulse cycle is called the Pulse Repetition Interval (PRI). The number of pulses transmitted per second is referred to as the Pulse Repetition Frequency (PRF), which is inversely related to the PRI.

2.1.2 Measuring range

The target range can be measured based on the round-trip-time at the speed of light. Since the signal has to travel to the target and back, the range is $2R$ and is found by

$$R = \frac{c\Delta T}{2} \tag{2.1}$$

where R is the target range, ΔT is the time between signal transmission and reception and c is the speed of light, at $c \approx 3 \times 10^8 m/s$.

A problem with pulsed radar occurs if the pulse round-trip travel time, ΔT, is larger than

the PRI. If a new pulse is transmitted before the previously transmitted pulse has returned, the received pulse could be a reflection of either one of them. This is known as a *range ambiguity*. To avoid range amgiguities, the PRI must be larger than the maximum pulse round-trip travel time, as stated in the condition below.

$$PRI \geq \Delta T_{max} = \frac{2R_{ua}}{c} \tag{2.2}$$

where T_{max} is the maximum round-trip time and R_{max} is the maximum target range that can be unambiguously achieved. Similarly,

$$R_{ua} \leq \frac{c \cdot PRI}{2} = \frac{c}{2PRF} \tag{2.3}$$

Symbol	**Definition**	**Unit**
R	Range	m
c	Speed of light	m/s
ΔT	Round-trip time	s
R_{ua}	Unambiguous Range	m
v_r	Radial velocity	m/s
f_r	Received Frequency	Hz
f_t	Transmitted Frequency	Hz
f_d	Doppler Frequency	Hz
f_0	Carrier frequency	Hz
λ	Wavelength	m
t	Time	s
A	Amplitude	V
θ	Phase	rad
ϕ	Shifted phase	rad

Table 2.1: List of common radar theory symbols

2.2 The Doppler Shift

Due to noise and clutter it can be hard to discriminate target signals. Modern radars take advantage of Doppler information to separate static clutter from moving targets. Measuring Doppler characteristics can also help distinguish several targets at the same range.

The Doppler frequency is the change of frequency between the transmitted signal and the

received signal due to a target in motion relative to the radar. The received frequency, f_r, can be explained by the theory of special relativity as in equation (2.4).

$$f_r = \frac{1 + v_r/c}{1 - v_r/c} f_t \tag{2.4}$$

where f_t is the transmitted frequency and v_r is the radial velocity of the target along the radars Line of Sight (LoS). A positive v_r represents an approaching target and will increase the received frequency. Similarly, a receding target with velocity $-v_r$ will decrease the received frequency.

Considering that radar targets will have radial velocities much smaller than the speed of light, the equation can be simplified. By expanding the denominator into a binomial series and discarding all but the first order term, leaves

$$f_r = [1 + 2(v_r/c)] f_t \tag{2.5}$$

The change in frequency, f_d, is called the *Doppler frequency* or the *Doppler shift* and is given by

$$f_d = f_r - f_t = \frac{2v_r}{c} f_t \tag{2.6}$$

As was shown for equation (2.4), a positive radial velocity causes a positive Doppler shift representing an approaching target, just like a receding target causes a negative Doppler shift.

2.2.1 Unambiguous Doppler shift

The Doppler shift is sampled at the PRF. If the PRF is not high enough, Doppler frequency amgiguities can occur. Nyquist's sampling criterion states that the maximum frequency that can be unambiguously measured is half the sampling frequency. In a radar, the Doppler shift frequency interval is therefore limited to

$$f_{d,max} = \pm \frac{PRF}{2} \tag{2.7}$$

where $f_{d,max}$ is the maximum unambiguous Doppler shift. Relating it to radial velocity reveals

$$f_{d,max} = \frac{2v_{ua}}{\lambda} \tag{2.8}$$

where $v_{ua} = \lambda \cdot f_{d,max}$ is the maximum unambiguous radial velocity and λ is the wavelength of the transmitted wave. If a velocity target violates Nyquist and exceeds $v_{ua}/2$ its echo will

alias into a new value $v' = v + k \cdot v_{ua}$ that falls within $\pm v_{ua}/2$.

By looking at equation (2.3) it is clear that the choice of PRF leads to a trade-off in maximizing range and frequency measurements. Maximizing unambiguous range requires lower PRFs whereas maximizing unambiguous Doppler shift requires higher PRFs.

2.2.2 The Micro-Doppler Effect

All objects in motion cause frequency modulations on the carrier signal. These motions that are not the bulk motion of the object are known as *micro motions*. That can be swinging arms or legs of a human or a rotating propeller of an aircraft. The superposition of these modulations create distinct signatures known as *micro-Doppler signatures*. Such signatures contain information about the kinematic properties of the target. This way, distinct target features can be extracted and used for classification purposes. [3]

2.3 Coherent Detector

Equation (2.6) shows the physical relationship between the Doppler shift and the radial velocity. To actually measure the Doppler shift, modern radars often apply a coherent detector. It measures the amplitude and phase of the received signal and downconverts it to baseband, where it is more easily processed. The transmitted RF signals at frequency f_t will from now on be referred to as being at the *carrier frequency*, f_0.

Consider a transmitted sinusoid at the radar's carrier frequency, f_0, at time $t = 0$:

$$x(t) = Acos(2\pi f_0 t + \theta) \tag{2.9}$$

This pulse can be characterized by its amplitude, A, and phase, θ. This can be seen by rewriting it into its exponential form:

$$x(t) = \Re(Ae^{j\theta}e^{j2\pi f_0 t}) \tag{2.10}$$

where $\Re$ represents the real part of the complex number. The returned signal from a target at range R_0 will be

$$
\begin{aligned}
x\Big(t-\frac{2R_0}{c}\Big) &= A'cos\Big[2\pi f_0\Big(t-\frac{2R_0}{c}\Big)+\theta\Big] \\
&= A'cos\Big[2\pi f_0 t+\theta-\frac{4\pi R_0}{\lambda}\Big] \\
&= \Re(A'e^{j\left(\theta-\phi\right)}e^{j2\pi f_0 t})
\end{aligned}
\tag{2.11}
$$

The returned signal in equation (2.11) has a new amplitude, A', a phase shifted by $\phi = -4\pi R_0/\lambda$ radians and a time delay by $\frac{2R_0}{c}$. A radar must measure these three parameters.

The coherent detector takes the sinusoid echo of equation (2.11) as input to two channels as depicted in figure (2.2). The upper channel is mixed with a reference cosine signal, typically the transmitted signal. The product of two trigonometric functions is a sum of two terms: a summation term and a difference term. The summation term is removed by the low pass filter, leaving the output $I(t) = A'cos(\theta - \phi)$, which is called the *in-phase* term or the *I-channel*. The lower channel is mixed with a 90 deg phase shifted reference signal $2sin(2\pi f_0 t)$. It outputs a signal which is *in-quadrature* with the upper channel and is therefore called the *Q-channel*.

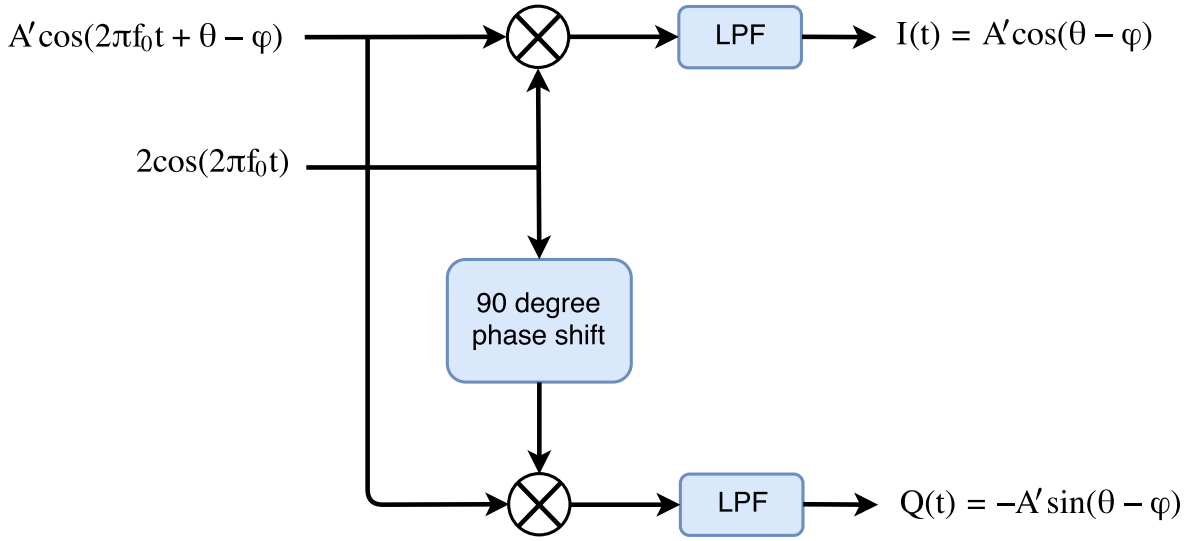


Figure 2.2: Coherent Detector

The Q-channel alone has ambiguities in finding the amplitude and phase. Along with the I-channel, it forms a complex output, $y(t) = I(t) + jQ(t)$, that resolves the ambiguities. For

f_0 and R_0 the output will be:

$$\begin{aligned} y[0] &= I[0] + jQ[0] \\ &= A'cos(\theta - \phi) + jA'sin(\theta - \phi) \\ &= A'e^{j(\theta-\phi)} \end{aligned} \tag{2.12}$$

For multiple such time samples the ouput will form a discrete-time signal $y[m] = I[m]+jQ[m]$.

The removal of the sum frequency terms by the lowpass filters is called *downconversion to baseband*, as it outputs a sinusoid at a frequency of zero Hz, i.e. a constant pulse.

2.4 Feature extraction

The process of extracting micro-Doppler information is covered in [6, 11] and will be repeated here, only with some changes of notation from [9] in consistency with the theory in this thesis. Figure (2.3) shows the steps of the process together with the final step of classifying the micro-Doppler data. The purpose of feature extracting is to obtain relevant information such as gait frequencies, gait amplitudes and object velocities. These are features that can prove important in the classification process.

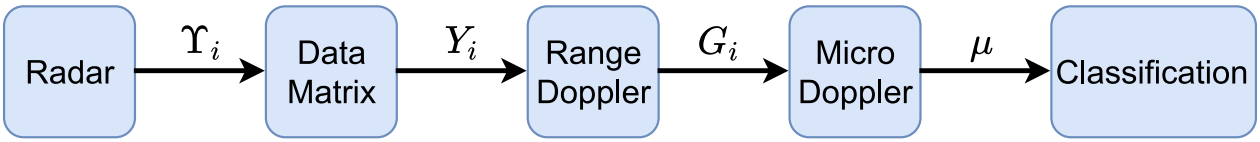


Figure 2.3: Feature extraction process

Symbol	Definition
L	Number of range bins per pulse
M	Number of pulses in time series
I	Number of data matrices
τ	Pulse length
T	PRI
Υ_m	Frame number m
G_i	Range Doppler number i
μ	Micro-Doppler matrix

Table 2.2: List of common symbols in the feature extraction process

2.4.1 Fast time/Slow Time

To measure the time delay, the receiver samples repeatedly after the pulse is transmitted. These time samples form complex samples from the I and Q channels that are stored in a vector of samples called *range bins*. For each pulse there are L such range samples. Since the range is sampled at a much higher rate than the pulses, which are sampled at the PRF, the range axis is called *fast time*. Similarly, the pulse number axis is called *slow time*.

Note that the terms pulse and frame will be used interchangeably. The term pulse has a more physical intuition to it, whereas frame represents the pulse in a mathematical form.

2.4.2 Pulse Radar Data Matrix

A single pulse of length τ has a Doppler resolution $1/\tau Hz$. This resolution can be improved by measuring a series of M pulses together, giving a Doppler resolution $1/(MT)Hz$, where T is the PRI. These M frames are stacked column-wise in a pulse radar data matrix, as illustrated in figure (2.4). Each frame, Υ_m, contains L range bins.

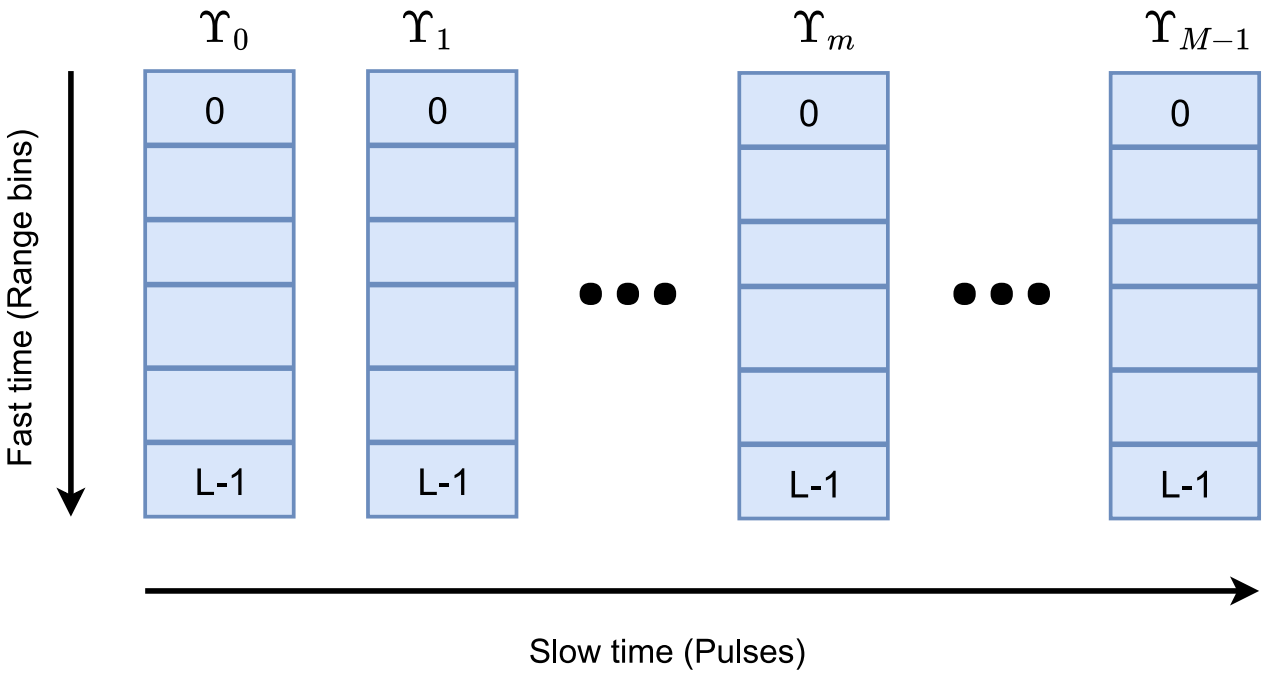


Figure 2.4: Data matrix

The data matrix, Y_i, is mathematically represented by:

$$\mathbf{Y}_i = \begin{bmatrix} y_{0,0} & y_{0,1} & \cdots & y_{0,M-1} \\ y_{1,0} & y_{1,1} & \cdots & y_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{L-1,0} & y_{L-1,1} & \cdots & y_{L-1,M-1} \end{bmatrix} \tag{2.13}$$

where $i \in [1, I]$ represents the i^{th} data matrix out of a total of I matrices. Each element in the matrix is given by

$$y_{l,m} = \Upsilon_{i+m-\frac{M}{2}}[n] \tag{2.14}$$

2.4.3 Range-Doppler Spectrum

Before Doppler information is extracted, stationary clutter around $0Hz$ is removed by subtracting the frame average over each range bin, as in equation (2.15).

$$\bar{y}_{l,m} = y_{l,m} - \frac{1}{M}\sum_{m=0}^{M-1} y_{l,m} \tag{2.15}$$

The Doppler spectrum for all range bins is the short-time Fourier transform (STFT) of the slow time data. In other words, for each range bin the Doppler spectrum is calculated across M frames. The result is a superposition of all contributions. The *Range-Doppler spectrum* is found by finding the Doppler spectrum across all range bins and is mathematically represented by:

$$\mathbf{G}_i = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,M-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{L-1,0} & g_{L-1,1} & \cdots & g_{L-1,M-1} \end{bmatrix} \tag{2.16}$$

G_i is the i^{th} range Doppler matrix, where $0 \leq i \leq I$. $g_{l,m}$ are Doppler spectrum elements (spectrograms) at specific range bins and Doppler frequencies. Equation (2.17) shows how to calculate these spectrograms by taking the magnitude squared of the STFT along with a window function, $w(t)$. Common window functions used are *Gaussian windows* or *Hamming windows*.

$$g_{l,m} = \left| \sum_{m=0}^{M-1} w(m)\bar{y}_{l,m} \exp\Big(- \frac{j2\pi km}{M} \Big) \right|^2 \tag{2.17}$$

2.4.4 Joint time-frequency analysis

To be able to analyse time-dependent frequency information, the Fourier transform is not sufficient. Joint time-frequency analysis is a commonly used method to describe signals in both the time and frequency domains.

To represent the range Doppler matrices in the joint time-frequency domain, the micro-Doppler matrix, μ, is created:

$$\mu = \begin{bmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,I-1} \\ u_{1,0} & u_{1,1} & \dots & u_{1,I-1} \\ \vdots & \vdots & \ddots & \vdots \\ u_{M-1,0} & u_{M-1,1} & \dots & u_{M-1,I-1} \end{bmatrix} \tag{2.18}$$

The range Doppler matrices are range gated around the range of interest. Each element, $u_{m,i}$, corresponds to the row-wise sum throughout the range gated range bins of the belonging matrix G_i. These operations result in a matrix in the time-frequency domain. The method is illustrated by figure (2.5), where the blue columns cover the range gate where the frequency content is summed over and collected.

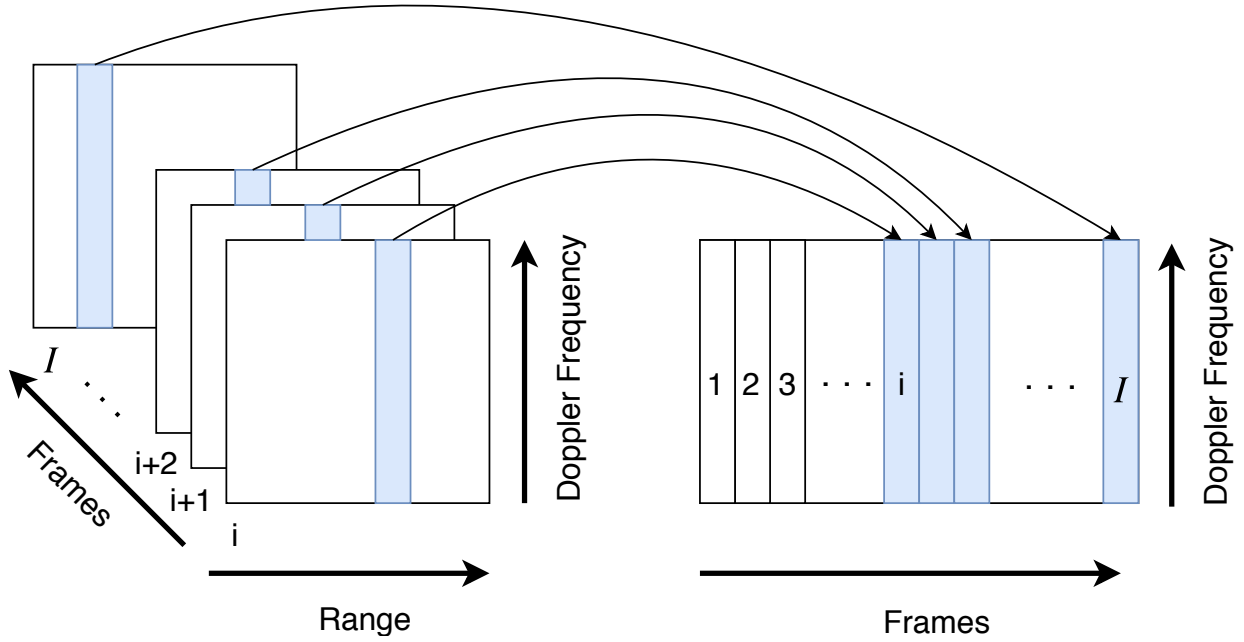


Figure 2.5: An illustration showing how a micro-Doppler matrix is constructed from range Doppler matrices.

Chapter 3

Neural Networks

The neural network theory is primarily based on Michael Nielsen's book *Neural Networks and Deep Learning* [8]. Exceptions are otherwise stated within the relevant paragraphs. A thourough review of the most important concepts is covered in [11]. Concepts necessary to grasp the basic understanding of CNNs are revisited and/or repeated here to provide sufficient theory for the classification methods conducted in this thesis.

Neural networks are mathematical models that learn from experience using observational data. This data is called training data and consists of pre-labeled data points. A properly trained neural network can give very high accuracy in applications regarding classification, such as speech recognition or image classification. These models consist of interconnected layers containing nodes known as neurons. In this chapter, two different types of neural networks are presented, namely *feedforward neural networks* and *CNNs*, where the former will serve as a basis for the latter.

3.1 Feedforward Neural Networks

3.1.1 Structure

An example of a feedforward network is illustrated in figure (3.1). This network consists of three layers: An input layer, a hidden layer and an output layer. The hidden layer is also called a fully connected layer, as its neurons connect to all neurons in the neighboring layers. There can be as many hidden layers in a network as is necessary. A network of many layers are called *deep networks*. These layers learn features of their inputs. The first layers pick up rough features, whereas deeper layers look for finer details.

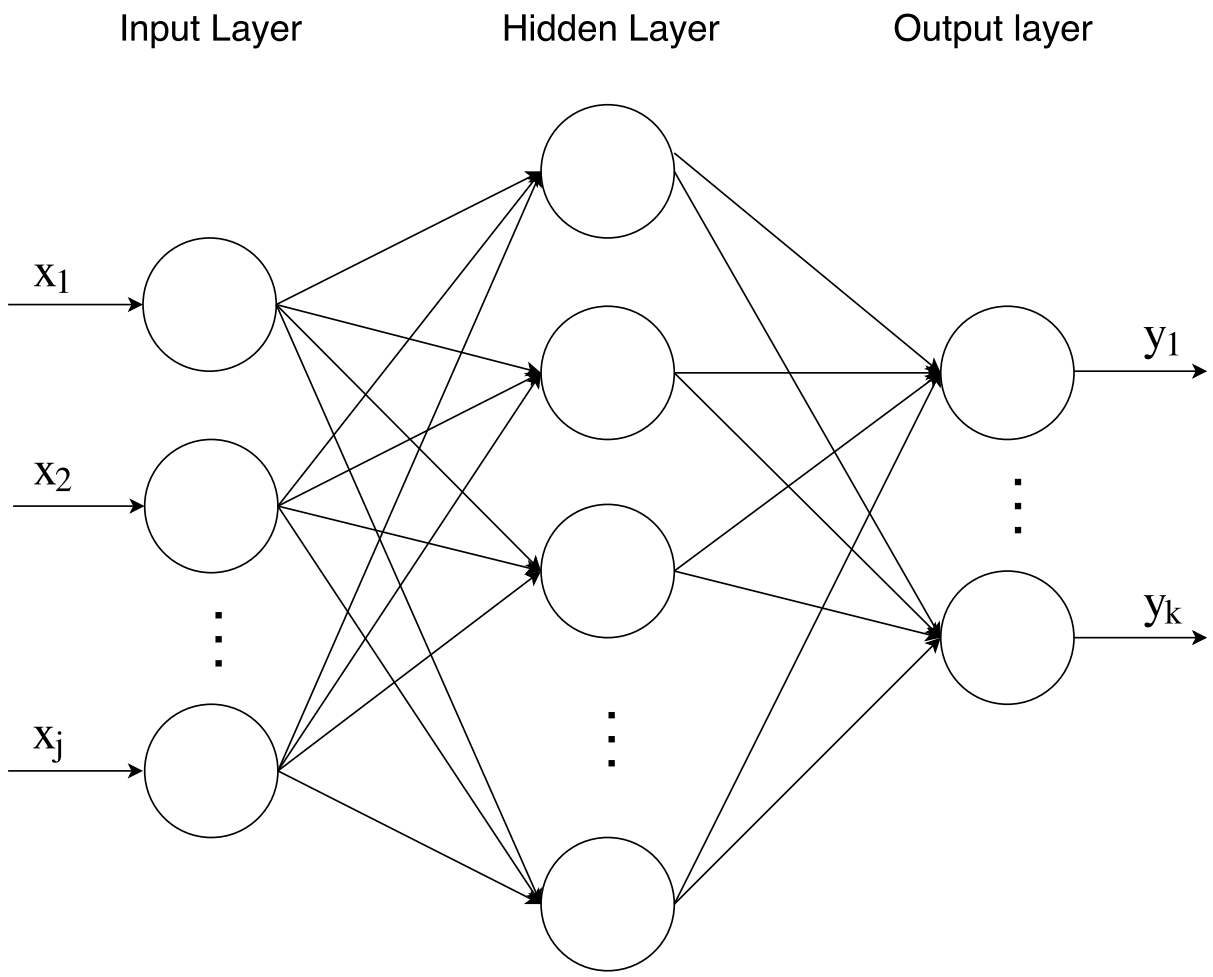


Figure 3.1: Illustration of a feedforward network [11].

The input to the network is the vector $x = [x_1 x_2 ... x_j]$, where j is the number of inputs. The inputs are propagated throughout the network, visiting all neurons, until the last layer wher the output $y = [y_1 ... y_k]$ is calculated. There are as many output neurons, k, as there are classes to be classified.

3.1.2 Activation functions

All inputs to the neurons are weighed by weights, w. By iteratively training the neural network, these weights are adjusted to eventually give importance to specific features. The goal of training is to adjust the weights in a way that minimizes the difference between the output of the network and the real output already provided in the labeled training data. The sum of all weighed inputs to the neurons at layer l are represented by the intermediate vector z^l, given by equation (3.1):

$$z^l = w^l a^{l-1} + b_l \tag{3.1}$$

where w^l is a matrix containing all weights into layer l, a^{l-1} is the vector of *activations*, or outputs of the *activation functions*, of the previous layer. The quantity b^l is a constant bias term.

Each neuron contain an activation function that calculates the output from the given inputs. It can be any mathematical function, but should contain certain characteristics, as their gradients are part of the iterative learning process. Activation functions are written in compact form as in equation (3.2):

$$a^l = \sigma(z^l) = \sigma(w^l a^{l-1} + b^l) \tag{3.2}$$

There are several common activation functions with different pros and cons, such as the *sigmoid function*, *hyperbolic tangent function*, the *Rectified Linear Unit (ReLU)* and the *softmax function*. The sigmoid function and hyperbolic tangent function have saturation issues when calculating their gradients. Learning slows down if a gradient saturates and becomes zero, which is the root of a phenomena known as *learning slowdown*. For that reason, the ReLU is often the preferred activation function. It is given by equation (3.3):

$$\sigma(z) = max(0, z) \tag{3.3}$$

The ReLUs don't suffer any learning slowdown when increasing the weighed input. Still, if the weighed input is negative the gradient disappears entirely. To handle this, a linear term can replace the zero in equation (3.3). This version is called a leaky ReLU. The choice of activation function is not trivial, however, work on image recognition has shown that the ReLU outperforms other activation functions.

Lastly, the softmax function, given by equation (3.4), is a popular choice of activation function in the last layer. It generates output values on probability form that are easily interpreted.

$$\sigma(z) = \frac{\exp z}{\sum_k \exp z_k} \tag{3.4}$$

The nominator is one of the output neurons and the denominator sums up over all the output neurons, normalizing the values into probabilities.

3.1.3 Cost function

A cost function, or loss function, is defined to quantify how well the network performs during training. It is a function that provides a measure of distance between the output of the network and the real output. The output of the last layer is the input to the cost function. Consequently, some activation functions are more suitable depending on what cost function

is used. This is shown by looking at two different cost functions: the *quadratic cost function* and the *log-likelihood cost function*.

Quadratic cost function

The quadratic cost function simply calculates the mean square error:

$$C(w,b) = \frac{1}{2n}\sum_x \left\|y(x) - a\right\|^2 \tag{3.5}$$

where n is the total number of training inputs, a is the the output when x is input, y is the correct class and the sum is over all training inputs x. The quadratic cost function serves as a reasonable cost function as the cost always will be positive. The cost is also easily interpreted. The training algorithm has done a good job if the cost is close to zero, $C \approx 0$. Similarly, a large cost means worse performance. It's derivative with respect to the weights is given by:

$$\frac{\partial C(w,b)}{\partial w} = (\sigma(z) - y)\sigma'(z)x \tag{3.6}$$

By looking at equation (3.6), it is seen that the derivative of the cost function is dependent on the gradient of the activation function. The same goes for the gradient with respect to the bias, b. This becomes a problem when using activation functions such as the sigmoid function, which gradient becomes very small for inputs that are close to 0 or 1. This learning slowdown can be addressed by using different variations of cost functions and activation functions.

Log-likelihood cost function

One way to address the learning slowdown problem is with the combination of the softmax function and the log-likelihood cost function. This solution is useful for classification problems, as the outputs are interpreted as probabilities. The log-likelihood function is given by

$$C(w,b) = -\frac{1}{n}\sum_x \sum_j y \ln a \tag{3.7}$$

where the first sum sums up over all inputs to the corresponding output neuron number j and the second sum sums up over all outputs. High probabilities, that is if the network has done a good job, will generate low costs close to 0, $C \approx 0$. If the network performs poorly the softmax provides low probabilities, generating high costs. By analyzing the gradient with respect to the weights in equation (3.8), it is clear that there are no dependencies on the gradient of the activation function, hence solving the learning slowdown problem. Similar

reasoning goes for the bias, b.

$$\frac{\partial C(w,b)}{\partial w} = \frac{1}{n}\sum_x \sum_j x(\sigma(z) - y) \tag{3.8}$$

3.1.4 Gradient Descent

The network learns by iteratively improving its weights and biases. It does so by minimizing the cost as a function of the weights and biases. It is essentially an optimization problem, and the learning algorithm used to optimize these parameters is known as *gradient descent*. This method updates the weights and biases by moving them in the negative direction of the gradient of the cost function. It comes in many variations, a few of which will be studied further in this section.

Batch Gradient Descent

The Batch Gradient Descent (BGD) method calculates the derivative of the cost function over the entire training set and subtracts it from the respective parameter, being either a weight or a bias. The update rules are given by:

$$\begin{aligned} w_j' &:= w_j - \frac{\eta}{n}\sum_{i=1}^{n} \frac{\partial C}{\partial w_j} \\ b_j' &:= b_j - \frac{\eta}{n}\sum_{i=1}^{n} \frac{\partial C}{\partial b_j} \end{aligned} \tag{3.9}$$

where η is the learning rate, a step size deciding how fast the algorithm should converge. It is important when tuning the neural network that the learning rate is set correctly. If it is too small it will render the algorithm to converge slowly towards the optimum. Too large and the algorithm may overshoot the optimal solution. n is the total number of training inputs, which is called a batch. Optimizing over the entire batch can be computationally costly for large datasets. For this reason, the Stochastic Gradient Descent (SGD) method and variations of it is a more common way to go.

Stochastic Gradient Descent

The SGD method differs from BGD in that it optimizes over a smaller part of the training set, called a *mini-batch*. Selecting random samples from the training data into mini-batches and computing the gradients speeds up learning. Let m be the number of samples, then $x_1, x_2, ..., x_m$ and $y_1, y_2, ..., y_m$ forms a mini-batch. The update laws for the weights and

biases are then as follows:

$$\begin{aligned} w_j' &:= w_j - \frac{\eta}{m} \sum_{i=1}^{m} \frac{\partial C_{x_j}}{\partial w_j} \\ b_j' &:= b_j - \frac{\eta}{m} \sum_{i=1}^{m} \frac{\partial C x_j}{\partial b_j} \end{aligned} \tag{3.10}$$

During one SGD run, an estimate of the entire training batch is given, which is not as exact as for the BGD, but by chosing the mini-batch size properly it will be good enough. When a neural network learns, mini-batches are chosen and trained upon until all tranining samples are exhausted. One such cycle is called an *epoch*. Proper training involves completing many such epochs.

Momentum-based gradient Gradient Descent

The SGD method struggles when the gradient is zero, which occurs in local minimas and saddle points. This can be addressed by adding a momentum term to equation (3.10), that still exists even though the gradient is zero. The update laws with such a term are given by:

$$\begin{aligned} w_j' &:= w_j + \mu v - \frac{\eta}{m} \sum_{i=1}^{m} \frac{\partial C_{x_j}}{\partial w_j} \\ b_j' &:= b_j + \mu v - \frac{\eta}{m} \sum_{i=1}^{m} \frac{\partial C x_j}{\partial b_j} \end{aligned} \tag{3.11}$$

where v is a running average of the gradients that can be viewed as a velocity term. The algorithm now steps in a weighed average direction of the velocity and the gradient. μ is a friction term that decays the velocity. This momentum based method solves the problems that the SGD method encounters in local minimas and saddle points.

3.1.5 Backpropagation

This section on backpropagation is taken in its entirety from [11] and explains how the gradient descent methods are made possible to work with neural networks.

To actually be able to use the gradient descent method the gradients have to be calculated. This is done using the backpropagation algorithm. Before the idea behind backpropagation is explained, some notation is necessary. w_{jk}^l is the weight between the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer. Similarly, a_j^l is the activation from the j^{th} neuron in the l^{th} layer.

The idea behind backpropagation is that a small weight perturbation Δw_{jk}^l causes a small

change Δa_j^l in the activation from the j^{th} neuron in the l^{th} layer. This will cause a small change in all the activation in the next layer, and so on. In the end, it will have caused a small change in the cost function $C(w, b)$. This propagation of changes can be tracked using the chain rule along the path of change. Since there will be many such paths, the total change in the cost function can be obtained by summing over all the paths, as is shown in equation (3.12).

$$\frac{\partial C_{x_j}}{\partial w_j} = \sum_{mnp...q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \tag{3.12}$$

In summary, the backpropagation algorithm propagates the error backwards throughout the network in order to find a small change in weights that decreases the cost.

3.2 Convolutional Neural Networks

This far, inputs to the neural networks must be presented on vector form. Classification of images thus requires a mapping down to one dimension, resulting in a loss of spatial structure. Nielsen uses three fundamental ideas to explain the concepts of CNNs, which architectures conserve the spatial structure of images. These ideas are presented in this section.

3.2.1 Local Receptive Fields

In contrast with feedforward neural networks, images can be inputted directly into a CNN. They are usually presented in a four dimensional manner, $[m, n, c, I]$, where m and n are the width and height of the images respectively. c is the number of channels, which for RGB images is three and grayscale images is one. The last number, I, is the total number of images.

Small filters, presented by Nielsen as *local receptive fields*, look for spatial structures in the images. Figure (3.2) shows a 5×5 filter in the upper left corner of a 16×16 input image. A dot product between all the pixels in the filter and a region of the image produces a neuron in the next hidden layer. By convolving the filter over the entire image, a complete mapping of dot products is created. The resulting hidden layer is therefore better known as an *activation map*.

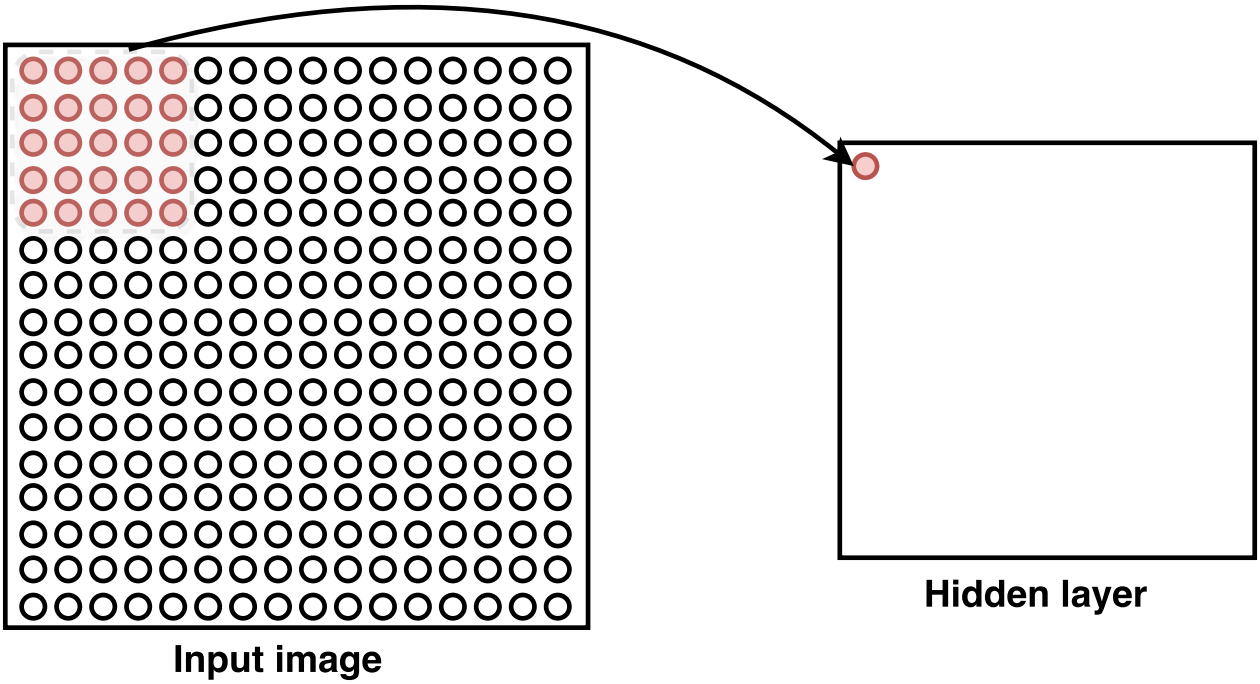


Figure 3.2: A local receptive field producing a neuron in the hidden layer. [11].

The 5×5 filter will with a stride length of 1 only be able to slide across the image eleven times until it reaches the image border. This results in a reduced activation map of size 11×11. A common solution to maintain the input size in the activation maps is to zero pad the borders of the input image.

3.2.2 Shared Weights

Each convolutional layer of the network can contain several such filters creating belonging actication maps. Each filter has weights corresponding to its pixels in addition to one bias term. That is, all neurons in an activation map has a set of shared weights and a bias. This means every filter will look for the same feature across the input image. If many filters are randomly initialized, a convolutional layer will look for correspondingly many features of the input images.

3.2.3 Pooling

To reduce the number of parameters throughout the network layers it is common to use *pooling layers*. They downscale the activation maps and exist in different variations. A widely used pooling method is called *max pooling*. A sliding window is slid across the activation map with a certain stride length and window size, outputting the maximum value within

the window. A window of size 2×2 reduces the activation maps by a factor of 2. Another variation is *L2-pooling*, which instead outputs the square of the sum of squares of each pixel within the window.

3.3 Measures against overfitting

Neural networks usually have a wide range of parameters (weights and biases). That means they can potentially describe training data down to the smallest detail. The learned model may agree well with the training data, however the network can be so specialized to the training data that when tested on different data, the results are not as good anymore. This is called *overfitting*. Such models can describe a given dataset, but are not generalized to handle other situations. There are several techniques to cope with the problem of overfitting. A solution can be to increase the training data, or even reduce the network. Acquiring training data is cumbersome, and large networks are more powerful than small networks. Therefore it is desirable to approach this differently [11]. Some common methods are discussed in the notes accompanying a deep learning class hosted by Stanford University [10], which are repeated here. The last section on training, validation and test data, chapter 3.3.3, is taken from [11] with minor modifications.

3.3.1 L2-regularization

A common form of regularization is to penalize the squared magnitude of all parameters. This is done by adding the term $\frac{1}{1}\lambda w^2$ in the cost function, where λ is the regularization strength. This way, large weights are penalized and smaller weights are preferred, meaning the network tries to use all its inputs rather than a small dominating portion.

3.3.2 Dropout

A simple, but effective regularization technique is *dropout*. During training, the neurons are kept active with some probability, p. This means, after training the modified network over a mini-batch of data, the weights and biases are updated and the process is repeated with a new subset of neurons kept active. This technique builds on a theory saying that different networks overfit in different ways. A kind of average scheme of different networks may be an efficient way of handling overfitting, which is what happens when training on different sub-networks for all mini-batches of data.

3.3.3 Training, validation and test data

It is common procedure to split the data into three sets: A training, validation and test set. The majority of the data should belong to the training set. Common distributions involve around 80% training data and 10% each of validation and test data, as shown in figure (3.3).

It is important that the data throughout the sets are Independent and Identically Distributed (IID). This assumption is common in statistics to simplify mathematical models. By independent it goes that the occurrence of one should not affect the occurrence of another. Identically means the distribution from which the data is drawn stays the same.

Figure 3.3: A dataset split into training data, validation data and test data.

The training set is the dataset which the network will use to learn from. The training set is labeled so that the network can move its weights and biases in the right direction.

During training, a separate labeled set can be used to check the performance. This set is called the validation set and can give indications on whether or not the network is overfitting to the training data. If the training loss is much higher than the validation loss then the network is overfitting. This is often done once or twice per epoch. The validation set does not contribute to training the network.

The test set is an unlabeled set used on a trained network to see if the network can classify data other than the training data correctly.

3.4 Weight initialization

Before the neural network can be trained, the weights and biases must be initialized. Weight initialization is a widely discussed topic with state of the art methods highly depending on network architectures. A common way to initialize the parameters is by drawing random weights from a zero mean Gaussian distribution with some standard deviation [5]. A problem occurs in deeper network where the models run into convergence troubles due to saturating weights throughout the network layers. A solution proposed by Glorot and Bengio [4] is using a scaled uniform distribution for initialization. This method is known as "Xavier" initializaion and assumes linear activation functions. When using ReLU functions this assumption

does not hold.

He, Zhang, Ren and Sun [5] proposes a robust initialization method that takes ReLUs into account. This method is known as "HE-initialization". It concludes that the variance of the weights should be given by

$$Var(W) = 2/n \tag{3.13}$$

where n is the number of input neurons. This leads to the randomly initialized weights with zero mean and standard deviation = sqrt(2/n). The bias terms are initialized to zero. This is according to [10] the current recommended initialization method.

Chapter 4

Methods

This chapter presents the methods used when acquiring data, both with Novelda X4 radar and by the use of the simulator developed in [7]. It also covers the design of the micro-Doppler images and the preparing of the datasets to be used as input data to the CNN. Finally, details of the CNN classifier as well as the design of the network architecture are presented.

4.1 Data acquisition

4.1.1 Real data

Real data was acquired over two turns in a controlled test environment in a film studio at Ila, Trondheim. The first round of recordings were performed in connection with the project work of the same subject in September 2017 [11]. The need of higher resolution images along with more data lead to a second round of recordings were some adjustments to the setup and radar parameters were made. This was carried out as part of this thesis in January 2017 at the same location.

The details on the first round of recording are repeated here. Data of a Weathen Terrier dog and two humans, walking with different gaits from different angles, was collected. This test setup is shown in Table (4.1). The recordings were video taped synchronously with the PRF, making analysis at specific frames possible. The test subjects carried a ranging device that estimated the range between the radar and the test subjects. These distances were provided alongside the recordings as csv files. A PRF of 300Hz was used so that the radar could capture high velocity targets, such as a running dog. The radar was placed at approximately 3m above ground. The angles at which the targets move relative to the radar is defined with 0 degrees pointing straight out from the radar.

Target	Distance	Direction	Movement style
Dog	2m	90°	Walking
Dog	4m	90°	Walking
Dog	-	0°	Walking
Dog	-	45°	Walking
Humans	-	45°	Walking, Marching, Hands in Pockets
Humans	1m	90°	Walking, Marching, Hands in Pockets
Humans	4m	90°	Walking, Marching, Hands in Pockets
Humans	-	0°	Walking, Marching, Hands in Pockets

Table 4.1: Test matrix round 1

A new round of recording was done to capture data with a higher SNR. The same film studio was used, only with a few adjustments to the setup. The radar was placed at a lower height so that the lobe pointed more directly toward the targets, allowing the capture of more stable radial velocities of the targets. The radar was mounted as shown in Figure (4.1), where the radar can be seen at the far right and the web camera in the middle.

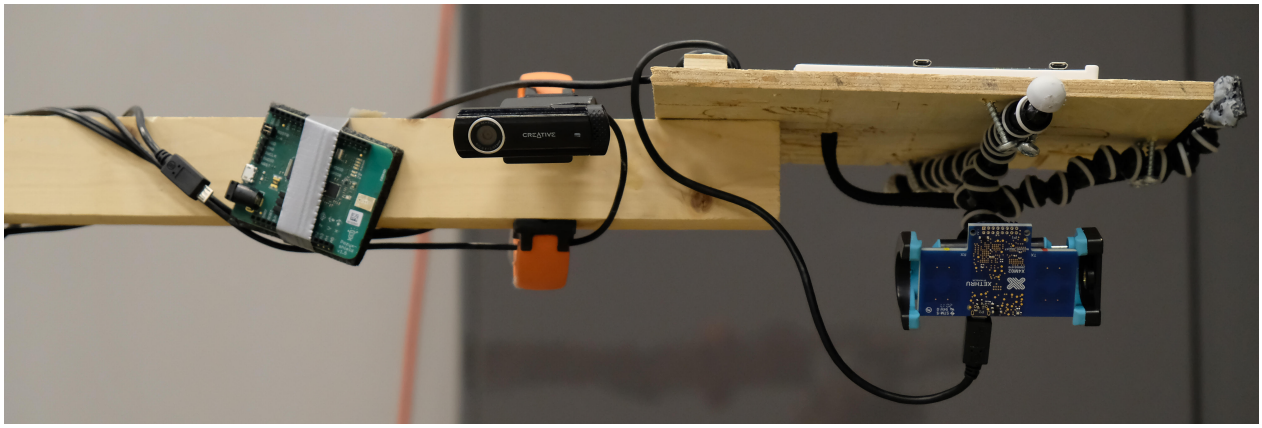


Figure 4.1: Test setup with mounted web camera and X4 radar.

A newly developed tracking algorithm replaced the ranger used earlier and so no form of ranging was done during recording.

Target	Direction	Movement style
Dog	Random	Walking
Dog	0°	Walking
Dog	45°	Walking
Dog	90°	Walking
Human	Random	Walking
Human	0°	Walking, Marching, Hands in Pockets
Human	45°	Walking, Marching, Hands in Pockets
Human	90°	Walking, Marching, Hands in Pockets

Table 4.2: Test setup round 2

Table (4.2) shows the test setup of the second round of recording. In contrast with round 1, only one human was recorded, as the pulse Doppler signatures of the two people contributing to round 1 had no prominent variations. Recording at 90 degrees was given lower priority as it is the angle at which the lowest change of radial velocity occurs. At the 90 degree recording, the distance was kept around 5m away from the radar. Separate recordings were made of the dog and a human walking around randomly. These were made to provide realistic test sets independent of the other recordings.

4.1.2 Micro-Doppler Radar simulator

Training neural networks usually require large datasets of IID data. Acquiring this manually is both an impractical and time consuming effort. For this purpose, a simulator was developed as part of another master thesis of spring 2018 [7].

It models humans and animals based on motion captures of humans and animals and creates simulated baseband data similar to that of the radar. The use of the simulator and the motion capture libraries enables signal capture of different size targets at different speeds and angles relative to the radar. This opens the opportunity for large and variate datasets that can be used for classification purposes.

4.2 Preparing datasets

The raw data acquired in chapter 4.1 is passed through the feature extraction process explained in chapter 2.4. For the resulting micro-Doppler signatures to be suitable as input data for the neural network, they must be processed into a certain format. The input images to the neural network are designed in a 4-dimensional fashion, containing the width, height,

depth and number of images. The width and height correspond to the time and Doppler frequency axes of the micro-Doppler matrices. This section covers the generation of input images and the preparation of the datasets used for training and testing the neural network.

4.2.1 Image design

Several design considerations must be considered when creating the micro-Doppler matrices and designing the images. Four important points to consider when processing the data are listed below and will be further discussed.

- PRF: Different target velocities require different PRFs to avoid aliasing.
- Window length: The choice of the window length, $w(t)$, in the STFT is crucial to present the matrices in a visually understandable manner.
- Time length: The time length in each input image decides how much gait information is included.
- Image size: The size of the images before being passed through the neural network stipulates the final image resolution.

Pulse Repetition Frequency

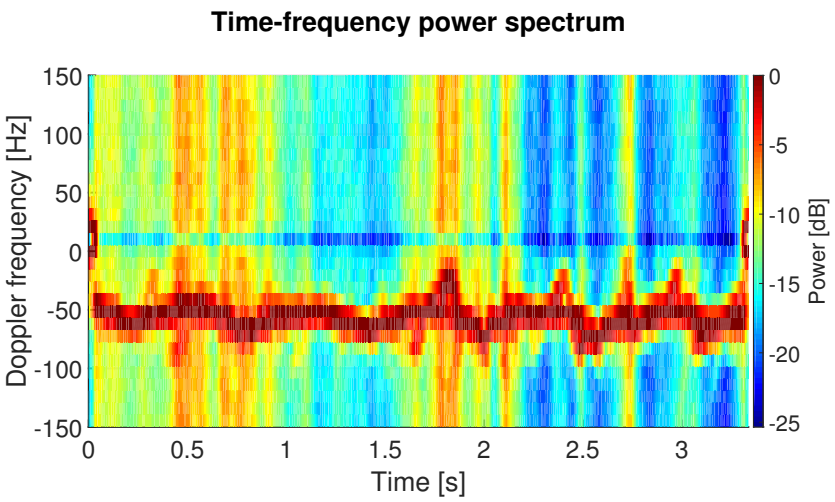
The use of the simulator enables large datasets to be created with the possibility to adjust the PRF to capture high target velocities. The data is generated using a $PRF = 2000$ except for the scenarios of running targets where a $PRF = 3000$ was necessary to avoid aliasing. As for the real data, the radar settings were set with a $PRF = 300$. This big gap in the PRF is due to X4-radar limitations. Most of the recordings from the second round, listed in table (4.2), contain target velocities exceeding the limit as given by the radar settings, thus rendering the data useless due to folding.

Window length

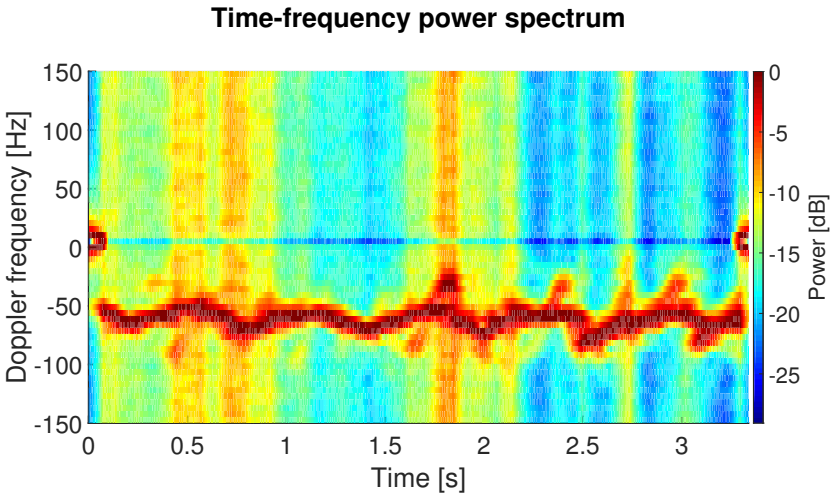
The number of elements, M, along the frequency axis of the micro-Doppler matrices is set by the window length, $w(t)$ multiplied with the PRF. A window too small results in a micro-Doppler signature with poor Doppler frequency resolution. If the window is too large the frequency content smears out into blobs with no visual meaning.

Figure (4.2) shows a micro-Doppler signature from the real dataset of a human walking towards the radar. It illustrates the effect of three different window lengths. In figure (4.2a) the window length of $w(t) = 0.1s$ is short enough to yield the micro-Doppler signature, only

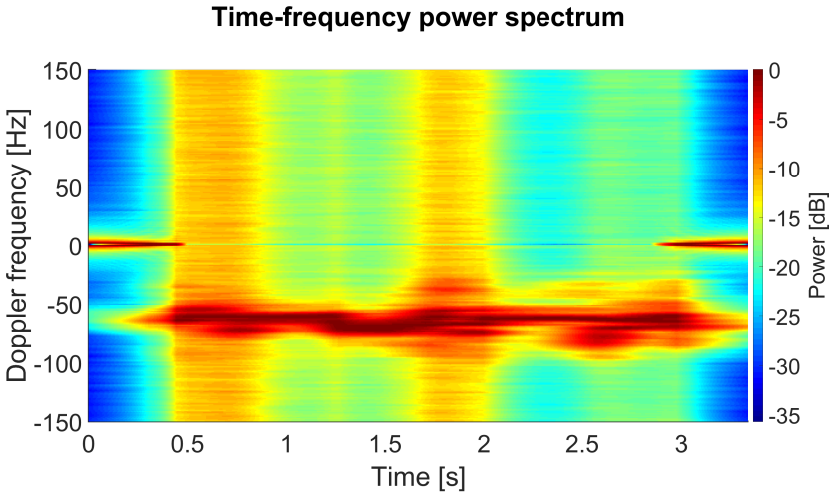
with poor resolution. Figure (4.2b) has slightly better resolution with $w(t) = 0.2s$, whereas the frequency content in figure (4.2c) is completely smeared out as the window length of $w(t) = 1.0s$ is too large. Due to these observations, the number of elements, M, along the frequency axis is set according to a window length $w(t) = 0.2s$ for both the real and simulated images.



(a) Window length 0.1s



(b) Window length 0.2s



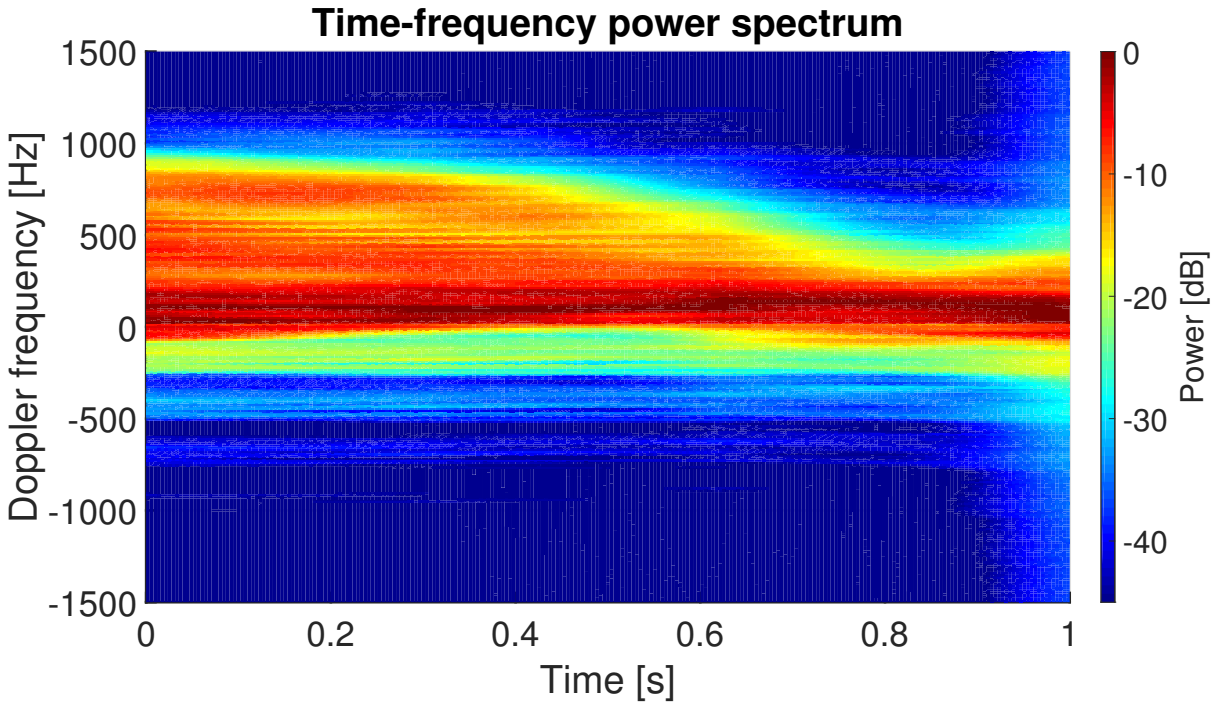
(c) Window length 1s

Figure 4.2: The effect of the STFT window length.

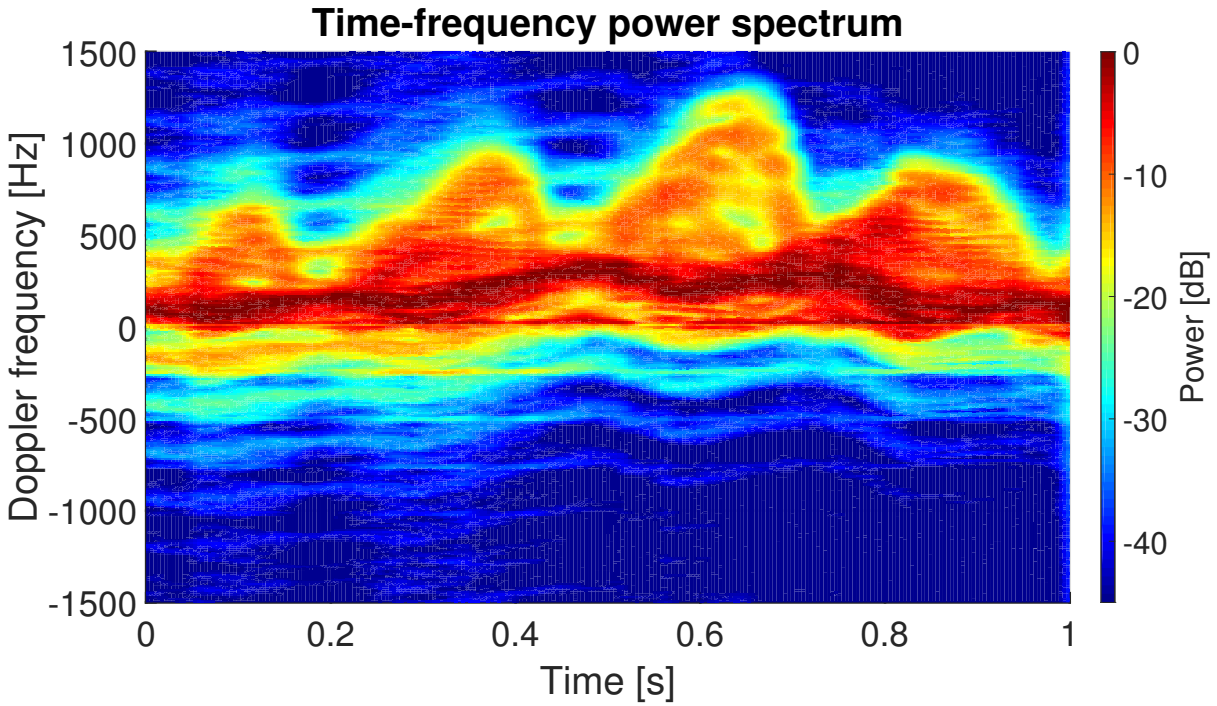
Time length

The time axis is set according to the number of frames, or range Doppler matrices, included in the micro-Doppler and the PRF. The longer the time interval, the more information, such as gait frequencies, are included in each image. In the earlier works of [2, 11], each micro-Doppler image was created by extracting fifty consecutive frames from a micro-Doppler signature. This allowed for large datasets of many different images, as every image only required fifty frames. However the resulting time length in each image is short and leaves small changes of oscillating frequency visible. An alternative is creating images capturing several gait cycles. Such longer time series images must be comprised of more range Doppler matrices and will therefore to some extent decrease the size of the dataset.

Figure (4.3) shows two images of different time lengths. Figure (4.3a) shows an image made to illustrate a design similar to that used in the earlier works of [2, 11]. Note that this example is made for the purpose of comparison and does not replicate the images used in the earlier works. It contains five hundred range Doppler matrices. Small changes in frequency can be seen, but not enough information is included to indicate the frequency at which limbs are oscillating. The image in figure (4.3b) shows an image containing three thousand range Doppler matrices. This results in an image containing several gait cycles, which gives a more complete conception of the target's locomotion style. This amount of frequency content over time is believed necessary in order to distinguish between dog and human micro-Doppler signatures and make good predictions.



(a) 500 range Doppler matrices



(b) 3000 range Doppler matrices

Figure 4.3: The effect of the time length.

To capture more frequency content in each image, time series comprised of 3000 range Doppler matrices, as in figure (4.3b), are used for the simulated data. With the PRFs used

when generating the data, this corresponds to time durations of $1.5s$ and $1.0s$. The real data is acquired with a lower PRF. Using 300 range Doppler matrices in each image thus creates images covering $1.0s$.

Input image size

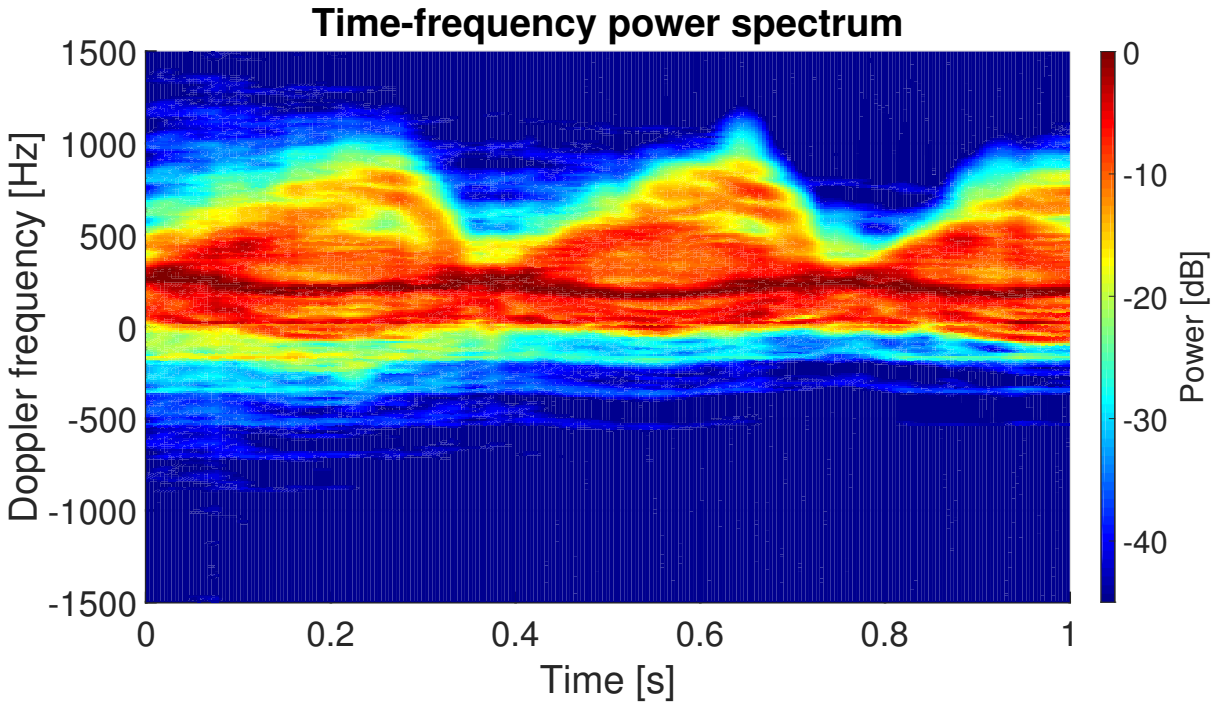
So far the simulated images are of size $M \times 3000$ and the real images are of size $M \times 300$. To limit the input size to the neural network, the images are downsampled using bicubic interpolation into quadratic images. Two different input sizes are tested upon. These are listed in table (4.3) along with their abbreviations.

Input size	**Abbreviation**
256×256	Im256
32×32	Im32

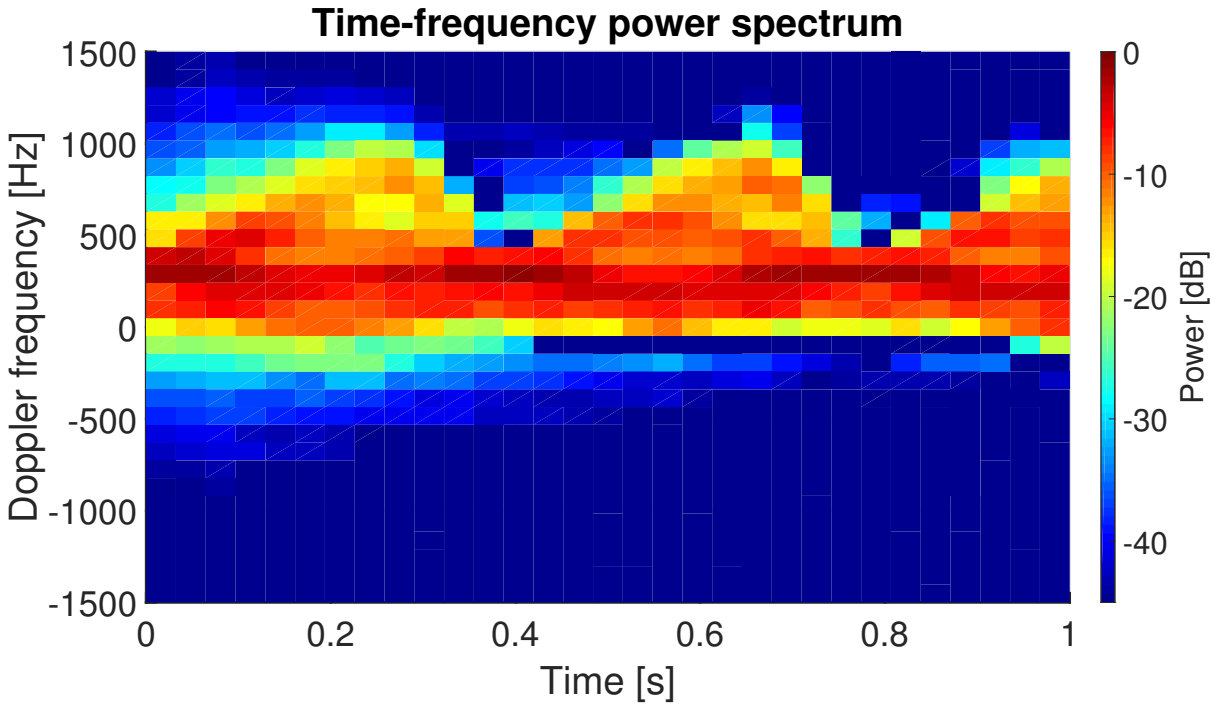
Table 4.3: Image input sizes and their abbreviations

The largest images are of size 256×256, keeping the resolution along the frequency axis for the simulated data. Further downsampling still results in images with key information visible to the eye, only with poorer resolution. Testing smaller size image decreases computation time. They contain much of the same information as the full resolution images, only with less noise. This might let the network focus on the important features and improve training.

Figure (4.4) compares the image types Im256 and Im32. They are taken from the same scenario of a simulated walking human. A brighter color indicates higher Doppler frequency power over the range bins at that point in time. Approximately two and a half gait cycles can be eyeballed in these images from the repetitive pattern due to swinging arms and legs and bulk motion. While the resolution of the Im256 image in figure (4.4a) is much higher than that of the Im32 image in figure (4.4b), the same pattern is still visible.



(a) Image type Im256



(b) Image type Im32

Figure 4.4: Input image with the signature of a walking human.

4.2.2 Input data

The complete dataset used for training is generated by the simulator and includes various situations. Table (4.4) lists the scenarios included in the training set. A total of 1560 training images are generated by extracting consecutive frames from the micro-Doppler signatures. As mentioned earlier, capturing several gait cycles in each image requires big portions of the original micro-Doppler signatures and leads to fewer images availbable for extraction. In order to compromise between capturing several gait cycles as well as creating a large dataset, all images overlap all but the 50 preceding frames. This results in a large set of data. The size of the dataset is ultimately limited by the access to motion capture data on dogs.

Class	**Situation**	**Radar placement**
Human	Walk	0°, 45°, 90°
Human	Walk fast	0°, 45°, 90°
Human	Walk exaggerated	0°, 45°, 90°
Human	Walk and turn	0°, 45°, 90°
Human	Run circles	-
Dog	Run	0°, 45°, 90°
Dog	Various motions	0°, 45°, 90°
Dog	Run and jump	0°, 45°, 90°
Dog	Run and jump onto box	0°, 45°, 90°

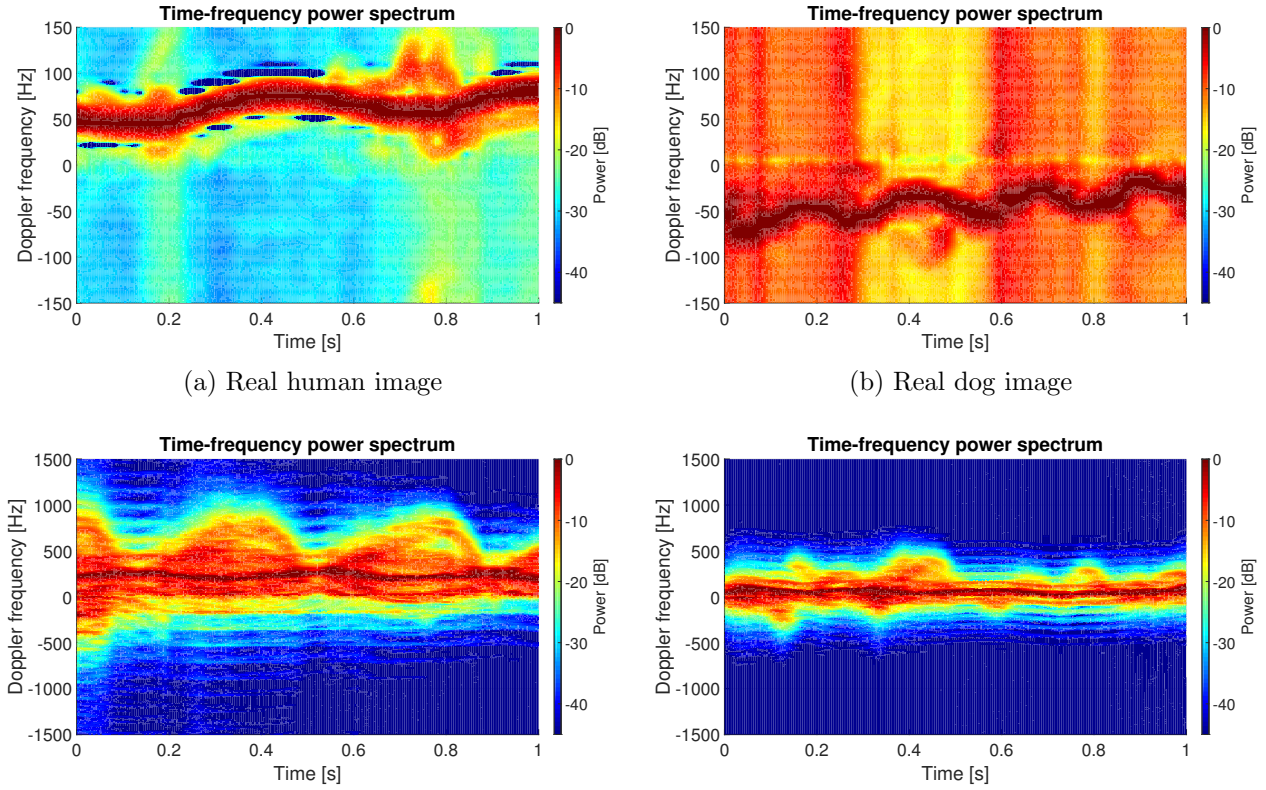
Table 4.4: Training data

A test set consisting of simulated data is created by extracting 10% of the training set in table (4.4). This portion is not used during training and is completely independent of the training data when tested upon. A second test set is generated from the real data listed in table (4.1). This set consists of 558 images. These test sets will be referred to as the simulated test set and the real test set.

Figure (4.5) compares human and dog images from both the simulated and the real datasets. Note that the vertical axes aren't the same in the simulated images as the real images. This is due to the different PRFs used when capturing the radar data. The number of image points are still the same as the final image sizes are set according to either image type Im32 or Im256. In this figure the images are of type Im256. Figure (4.5a) shows a micro-Doppler signature of a human from the real dataset. The positive Doppler frequencies indicate the human is walking towards the radar. Approximately one and a half period of the swinging pattern is visible. The dog has a higher swinging frequency, as seen in the dog image in figure (4.5b) with almost five periods visible. This makes sense from the standpoint of dogs

having shorter legs and higher gait frequencies. The negative Doppler frequencies means the dog is moving away from the radar. The frequency at which the objects move their limbs is an important feature that can possibly be used to separate the classes. As seen in both of the images coming from the real datasets, there is a large amount of noise visible. This is non-existent in the simulated images.

Figure (4.5c) shows a human image from the simulated dataset. The frequency patterns are clearly visible with almost three periods visible. The amplitude of the oscillations are much greater in the human image than what is the case with the simulated dog image in figure (4.5d). This indicates that the simulated human reaches higher velocities as it swings its limbs than the simulated dogs. Some weak oscillations can be seen in the dog image, however most of the frequency content stays near the bulk frequency. This difference in thickness of the signatures may be a feature that helps distinguish the image classes.



(a) Real human image

(b) Real dog image

(c) Simulated human image

(d) Simulated dog image

Figure 4.5: Human and dog images from simulated and real datasets

There are great differences between the simulated and real images when it comes to presence of noise and characteristics of the oscillating frequency patterns. The real images are marked by noise in the form of clutter from surrounding objects and propagation loss. The

propagation loss renders signals returning from a distance weaker. That is, the signal is weaker the longer it has travelled, making information of interest less visible. The vertical patterns arising in figure (4.5a) and figure (4.5b) are likely due to this phenomena.

The real data signatures are made more distinct by converting the data into a logarithmic scale. Noise can be removed by setting a decibel limit, allowing only frequency spikes of a certain power level to be visible. This effectively increased the contrast between the frequency patterns and the noise floor.

A neural network will be trained on simulated data and tested on both the simulated test set and the real test set. It is expected that the results on the simulated test set are better than on the real test set as only simulated data is trained upon. If classification can be done on simulated data it is likely it can be done with real data as well under the right circumstances. This implies that training is done properly with real data included in the training process and that the access to real data is good enough to create a large set of diverse images. It is however interesting to see if a network trained on simulated data can recognize features in a real test set.

Lastly, it was experienced during testing that an unbalanced training set may lead to the results being heavily biased. Rather than learning to identify important features of the data, a neural network may base its classification on the distribution of each class in the training data. Therefore the dataset is balanced such that there are an equal amount of dog images as there are human images.

4.3 Neural Network Classifier

The CNN is implemented using the "Neural Network Toolbox" in Matlab. Training is performed using an NVIDIA TITAN X graphics card (GPU), significantly decreasing computation time. The design is partly based on experience from the work in [11], where a smaller network consisting of fewer convolutional layers is studied. Other considerations are taken in line with state of the art theory from [10]. The CNN developed in this work is of a greater size than what was tested in [11], meant to capture a bigger set of features.

4.3.1 Neural network architecture

The input images are passed through five convolutional layers, each followed by a max pooling layer and leaky ReLU functions. The max pooling layers effectively downsamples the input images by a factor of 2 in each layer. To keep the computational complexity in each

layer the same, the number of filters are consecutively increased by a factor of 2 with the purpose of giving uniform importance throughout the network layers.

The last activation maps are mapped onto a fully connected layer, representing the strongest feautures of the input image. Several such layers gradually downsample the features towards the two classes. The features are weighed and computed into probabilities in the output layer by the softmax function. The output layer consists of two neurons representing a human and a dog. The probabilities in the output neurons are fed into the log likelihood cost function so that a cost can be calculated and used to update the weights, as described in chapter 3.1.3.

An illustration of a neural network architecture is shown in figure (4.6). It is based on the Im256 images. The dimensions in the activation maps changes with other input image types, but the general layer structure stays the same.

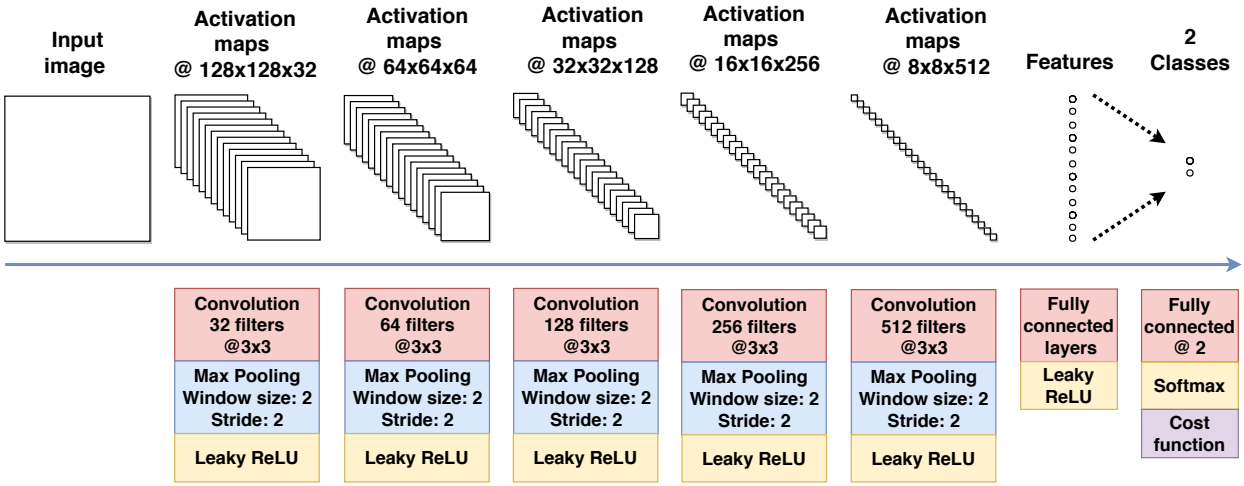


Figure 4.6: Neural network architecture.

The boxes underneath each layer contains layer parameter considerations and are color coded. The red box belongs to the respective convolutional layer and holds the number of filter and their sizes. The blue box contain information about the pooling layers. The yellow box tells which activation function is used and the purple box in the output layer is the cost function at the very end of the architecture, which here is the log-likelihood function.

4.3.2 Number of epochs and training iterations

The number of epochs is chosen such that network is trained enough for the loss and accuracy to stabilize. 20 epochs were trained upon when testing the Im256 images, whereas 50 epochs when testing the Im32 images. This was due to some of the configurations needing more

time to converge.

The mini-batch size is set to 128. With the total of 1560 images being split into 1248 training images and 312 validation images, this resulted in $floor(1248/128) = 9$ iterations per epoch. The Im256 images are therefore trained over a total of 20 epochs $\times$ 9 iterations = 180 iterations. Similarly, the Im32 images are trained over 50 epochs $\times$ 9 iterations = 450 iterations.

4.3.3 Initial learning rate

The learning rate in the SGD method is set on a piecewise drop schedule, starting at 0.1 and dropping with a factor of 10 every time one fifth of the training epochs are completed. This means the learning rate dropped every 4th epoch for the Im256 images being trained for 20 epochs and every 10th epoch for the Im32 images training for 50 epochs.

Such a piecewise schedule was shown to produce better results than training with a constant learning rate. This way it takes larger steps towards the optimal minimas in the beginning before it starts fine searching.

Chapter 5

Testing and results

This chapter covers the testing of the CNN and the input images as presented in chapter 4. The first section presents details explaining the test procedure and how the results are treated. The following sections present some selected results from testing with the two image types, Im256 and Im32, where the effects of L2-regularization and dropout are studied.

The major focus is on the simulated dataset as it was the main source of data and most importantly, the data comprising the training set. Three factors are studied and compared to analyze the performance of the network on the simulated data:

- Classification accuracies: The percentage of correctly classified test images.
- Training and validation loss: Loss curves help evaluate the model. Together the training and validation loss can give information about how the model fits to the data.
- Bar plots with predictions on test images in probability form.

These three factors give a combined evalutation of the network's performance. The filters of the convolutional layers can also be studied for some selected results in appendix A.

Classification accuracies on the real data are also presented to show how the network performs on real data which is not included in the training set.

5.1 Test procedure

The neural network as well as all the code for preprocessing data and testing the network are implemented in Matlab. Training information, such as the hyperparameters used, filters of the convolutional layers and loss curves, were saved to a simultaneously created and timestamped folder in a predefined path. Test results such as bar plots of the predictions on the

test data, confusion matrices and test accuracies on the simulated and real datasets were also saved. This way it was trivial to go back and look at the results from training.

The code below is a simplified version of the main file used when training and testing the network. The "trainNetwork" function is part of the neural network toolbox in matlab, whereas the other functions are custom made.

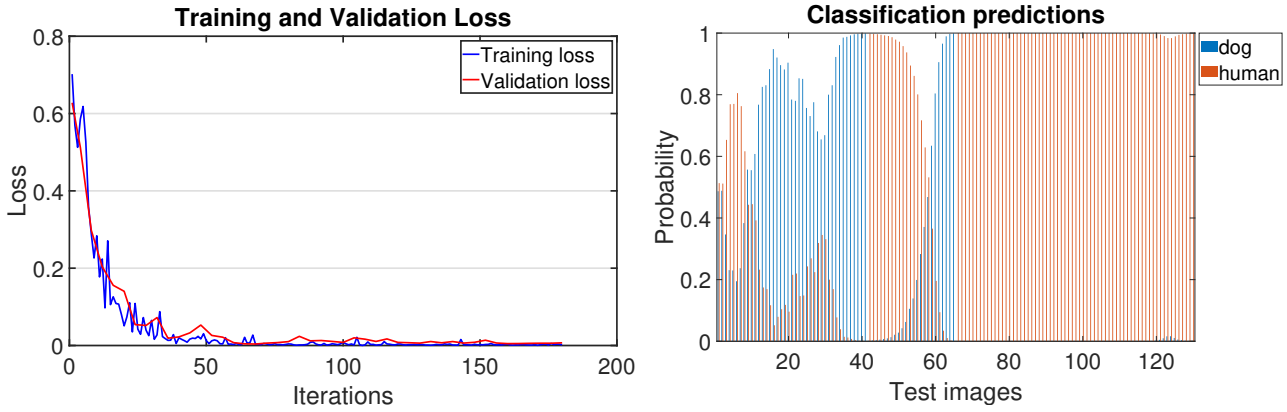
```
load(trainData);
load(trainLabels);
load(testData);
load(testLabels);

loadhyperparameters;
rng(seed);

layers = setLayers(hyperparameters);
layers = setWeights(layers,hyperparameters);
options = setTrainingOptions(hyperparameters);

[net,info] = trainNetwork(trainData,trainLabels,layers,options);

testNetwork(net, testData, testLabels, savePath);
saveTrainingResults(net,info,hyperparameters, savePath);
```

First, the preprocessed training and test data are loaded. The hyperparameters are then loaded as a struct. This struct keeps the parameters, such as initial learning rate, regularization strength, mini-batch size to mention a few, organized in an orderly manner. A seed is set by the "rng(seed)" command to produce predictable random weights and data shuffling. The layers are defined by "setLayers" and "setWeights" before the training options are set according to the hyperparameters. The layers and training options now define the neural network, which is trained by "trainNetwork" with the loaded training data. The "net" output argument is the trained network where the layers and the weights are defined. "info" contains information from the training process such as training loss and accuracy data. Finally, the network is tested and the results are saved to the predefined save path.

5.2 Testing on Im256 images

In this section the Im256 images are studied. With no form of regularization or dropout the network scored a classification accuracy of 81% on the simulated data and 50% on the real

data. The corresponding training and validation loss curves are shown in figure (5.1a). Both the validation loss and the training loss converges nicely towards zero. The validation loss follows the training loss for the most part, suggesting the network managed not to overfit to the training data. The predictions in figure (5.1b) shows that the network classifies all human images with high confidence, whereas portions of the dog images are incorrectly classified as human and in general classified with poor confidence. This result is used as a reference to the proceeding results when testing with regularization and dropout techniques.



(a) (b)

Figure 5.1: Loss curves and classification predictions on Im256 images.

5.2.1 Effect of L2-regularization

The L2-regularization method is tested with different regularization strengths, λ, to map the effects on training. Table (5.1) lists the classification accuracies on simulated and real test data resulting from test runs with different extents of regularization. Corresponding training and validation loss against training iterations are plotted in figure (5.2). Bar plots containing the network's predictions on the simulated test data are shown in figure (5.3).

Im256		
λ	**Simulated data accuracy**	**Real data accuracy**
1.0	59%	45%
0.1	100%	40%
0.01	89%	50%
0.001	73%	45%
0.0001	70%	50%

Table 5.1: Classification accuracies given regularization strengths tested on Im256 images.

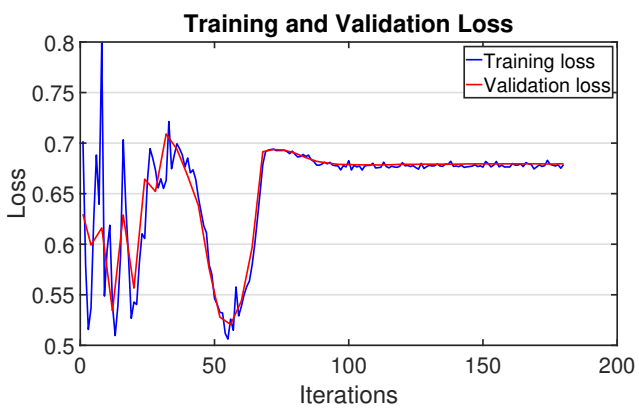
The highest classification accuracy of 100% is according to table (5.1) occurring with a regularization strength $\lambda = 0.1$. The loss curves stagnate around 0.1 in figure (5.2b). Spikes in the curves arise around the fortieth iteration before they converge towards the terminal loss. This can indicate that the SGD method has problems finding local minimas when optimizing the cost function with a regularization strength this high. By looking at figure (5.3b) it can be seen that all images were predicted correctly, only with little confidence as the network classifies most of the dog images with probabilities around 0.5. The human images are more easily classified with probabilities around 0.8.

A worse scenario occurs with $\lambda = 1$, which has a classification accuracy of 59%. The corresponding training and validation losses in figure (5.2a) stagnate at approximately 0.69. A loss of 0.69 is a typical value when using the log-likelihood cost function on two-class problems with probabilities equal to chance. The loss is calculated from equation (3.7) as $C = -ln(0.5) \approx 0.69$. The classification accuracy of 59% is a little higher. The predictions shown in figure (5.3) shows that all the test images are classified with probabilities close to 0.5, meaning the network classifies the images with little confidence. This discrepancy between the classification accuracy and the loss curves may be explained by the network's prediction levels not being exactly 0.5.

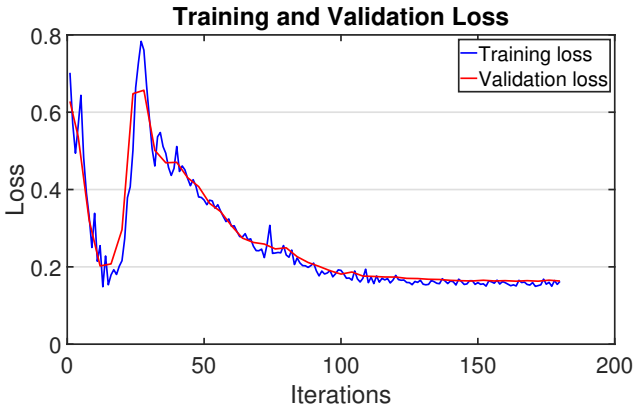
A $\lambda = 0.01$ resulted in a classification accuracy of 89%. The loss curves in figure (5.2c) converges smoothly to zero. Figure (5.3c) shows that the confidence of classification is higher than the previously cases presented, but still with a big portion of the dog images predicted with low probabilities. The incorrectly classified images are dog images being predicted as human.

The results achieved with $\lambda = 0.001$ and $\lambda = 0.0001$ are of slightly worse performance with accuracies 73% and 70%. The loss curves of figure (5.2d and 5.2e) show convergence towards zero for both cases. The test image predictions are quite similar as seen in figure (5.3d) and figure (5.3e).

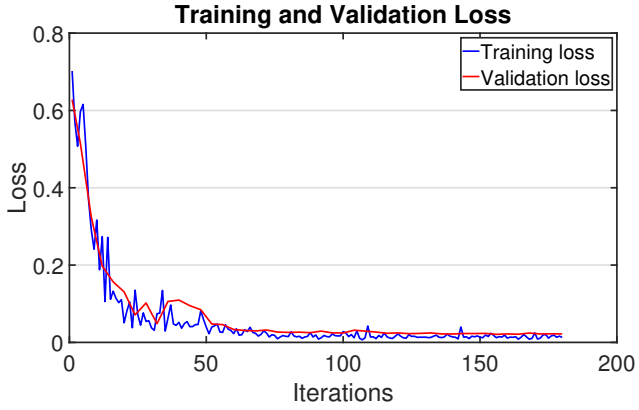
Poor performance is achieved throughout all regularization runs when testing on the real dataset. The best classification accuracy of 52.5% occurs with $\lambda = 0.1$.



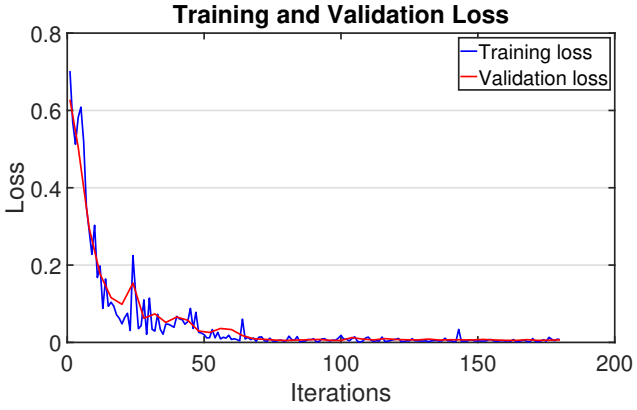
(a) $\lambda = 1$



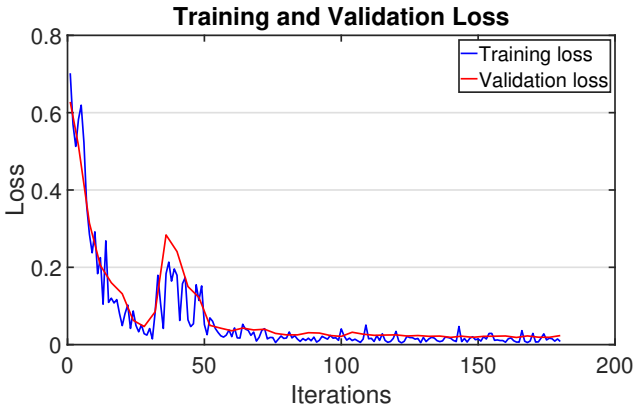
(b) $\lambda = 0.1$



(c) $\lambda = 0.01$

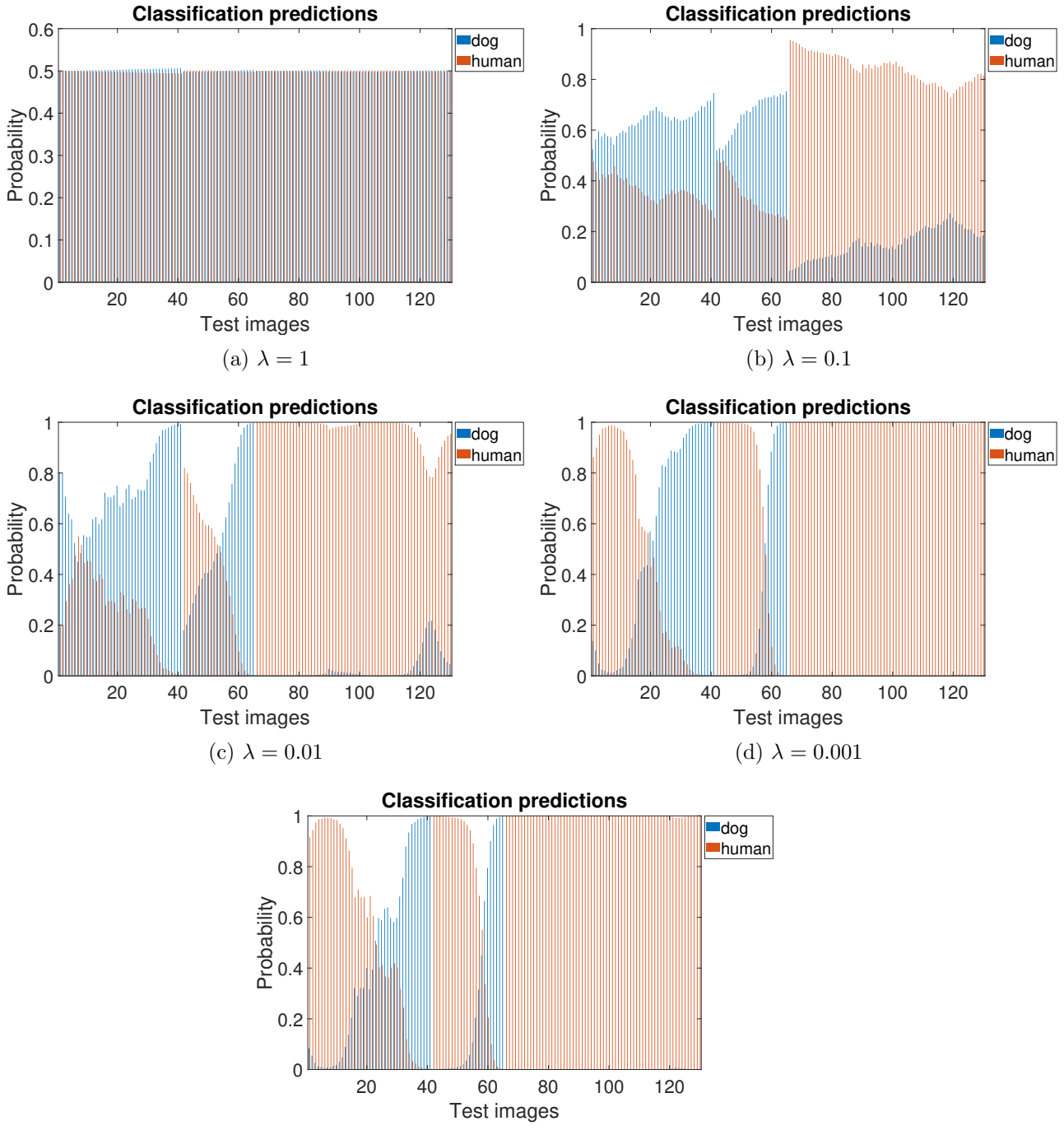


(d) $\lambda = 0.001$



(e) $\lambda = 0.0001$

Figure 5.2: Training and validation loss curves from regularization test runs.



(a) $\lambda = 1$

(b) $\lambda = 0.1$

(c) $\lambda = 0.01$

(d) $\lambda = 0.001$

(e) $\lambda = 0.0001$

Figure 5.3: Predictions on simulated Im256 images from regularization test runs. The 65 bars to the left are dog predictions and the 65 bars to the right are human predictions.

5.2.2 Effect of dropout

The dropout technique is tested with different dropout probabilities, p, as explained in chapter 3.3.2. The classification accuracies achieved are listed in table (5.2), with corresponding loss curves in figure (5.4). The network's predictions on the simulated data are presented in figure (5.5).

Im256		
p	**Simulated data accuracy**	**Real data accuracy**
0.1	78%	48%
0.2	70%	50%
0.3	84%	50%
0.4	61%	51%
0.5	62%	49%
0.99	50%	53%

Table 5.2: Classification accuracies given dropout probabilities tested on Im256 images.

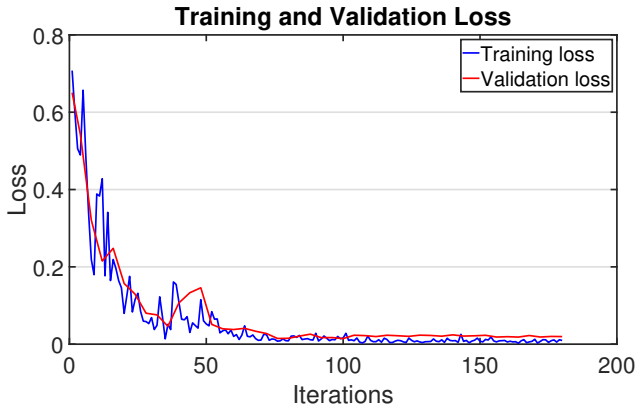
As seen in table (5.2), training the network with $p = 0.1$ and $p = 0.2$ resulted in 78% and 70% of the simulated test images being correctly classified. The loss curves in figure (5.4a) and figure (5.4b) converge smoothly towards zero. The predictions of $p = 0.2$ in figure (5.5b) are somewhat stronger than the network's predictions with $p = 0.1$ in figure (5.5a).

The highest classification accuracy on the simulated data was 84%. It was achieved with a dropout probability of $p = 0.3$. This is a slight improvement to the test run without any form of regularization or dropout. The loss curves in figure (5.4c) have spikes arising around the thirtieth iteration. After that they reach zero quickly, meaning the network eventually predicts the training data well. The validation loss follows the training loss which suggests the network manages not to overfit to the training data. The predictions in figure (5.5c) are strong on the human data. Some of the dog images are falsely or poorly predicted.

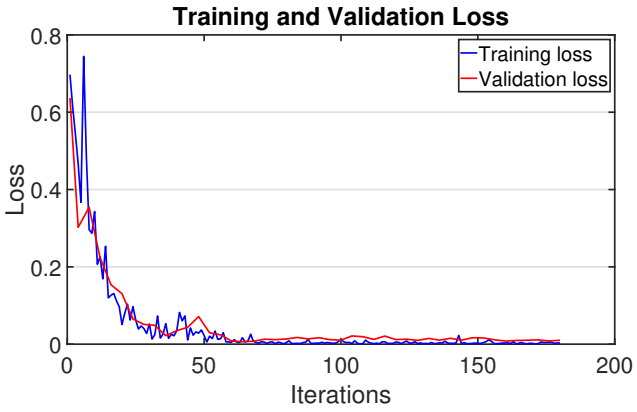
Further increasing the dropout probability results in lower accuracies. $p = 0.4$ scores a classification accuracy of 61% on the simulated data. The loss curves in figure (5.4d) has some troubles decaying in the first 20 iterations, but starts converging towards zero loss eventually. Figure (5.5d) shows that most images are predicted as human. A similar case arises with $p = 0.5$, which scores a classification accuracy of 62%. The predictions in figure (5.5e) are very similar to those of $p = 0.4$. The loss curves in figure (5.4e) reaches zero a little faster and are somewhat smoother.

Poor results are achieved with all the different dropout probabilities when testing on the real data. The classification accuracies are close to chance.

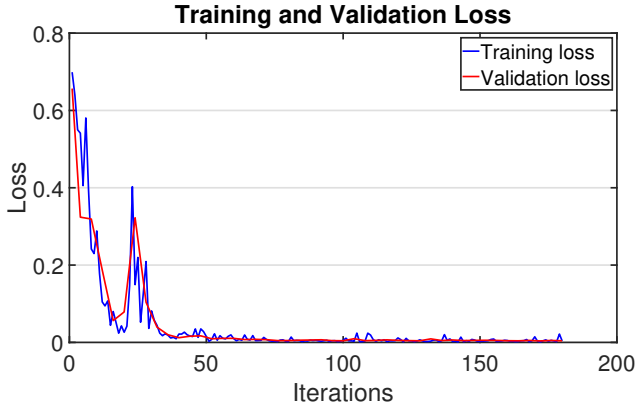
A sanity check was performed with a dropout probability $p = 0.99$ to see if the network behaves normally. This effectively killed 99% of the neurons during training and lead to 50% accuracy on the simulated test set and 53% accuracy on the real test set. The loss curves are shown in figure (5.4f). The loss requires many iterations before it starts converging. Neither the validation loss nor the training loss converge all the way to zero, meaning the network never fully manages to classify the training and validation images with high probabilities. Figure (5.5f) shows that the network consistently predicts the simulated test images as human. This result supports the sanity of the network.



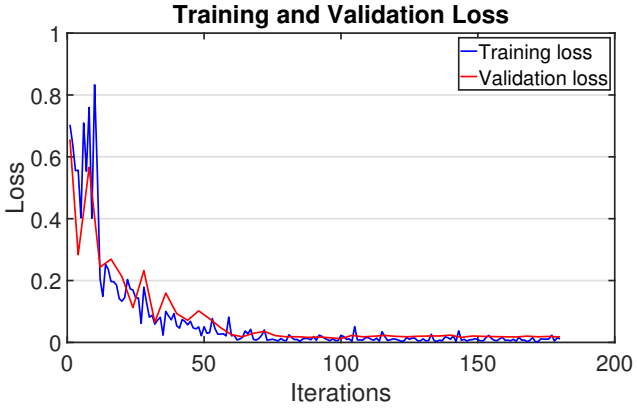
(a) $p = 0.1$



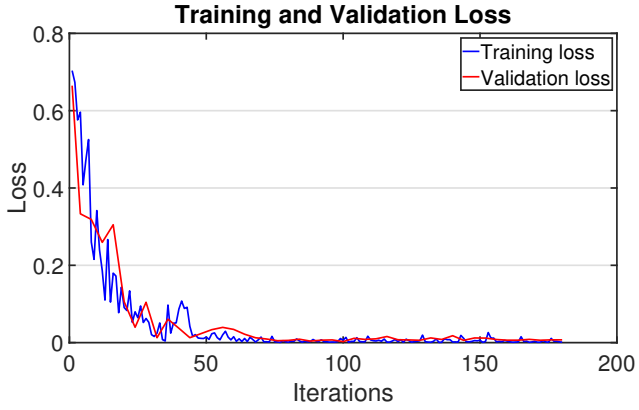
(b) $p = 0.2$



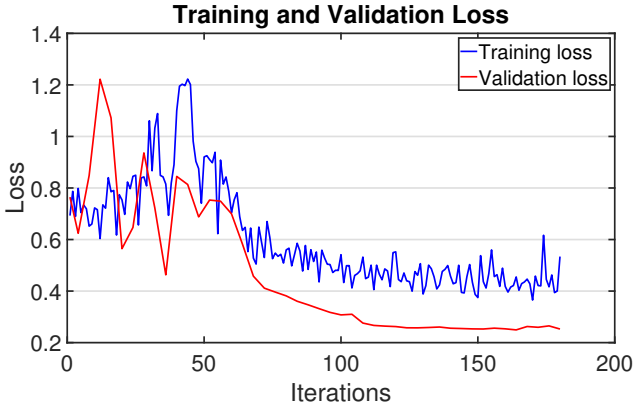
(c) $p = 0.3$



(d) $p = 0.4$

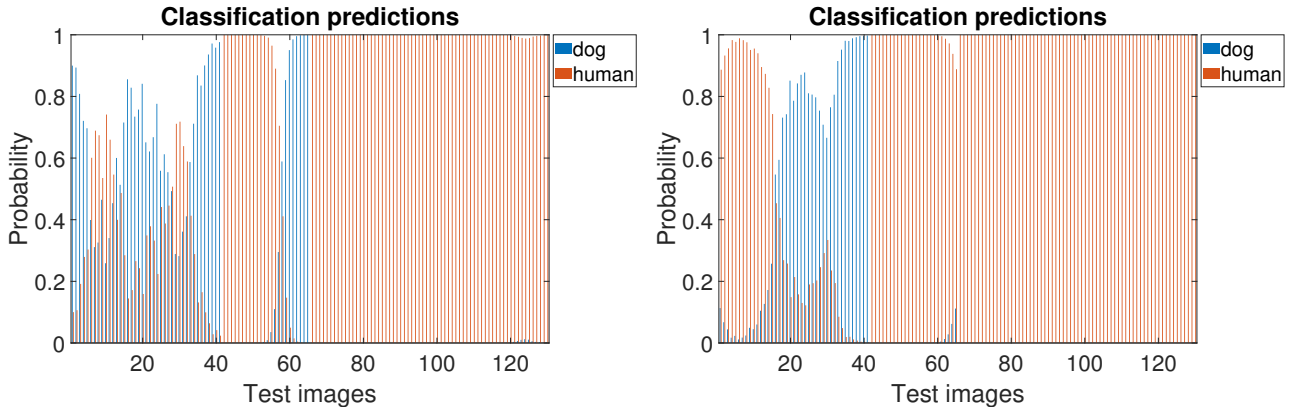


(e) $p = 0.5$



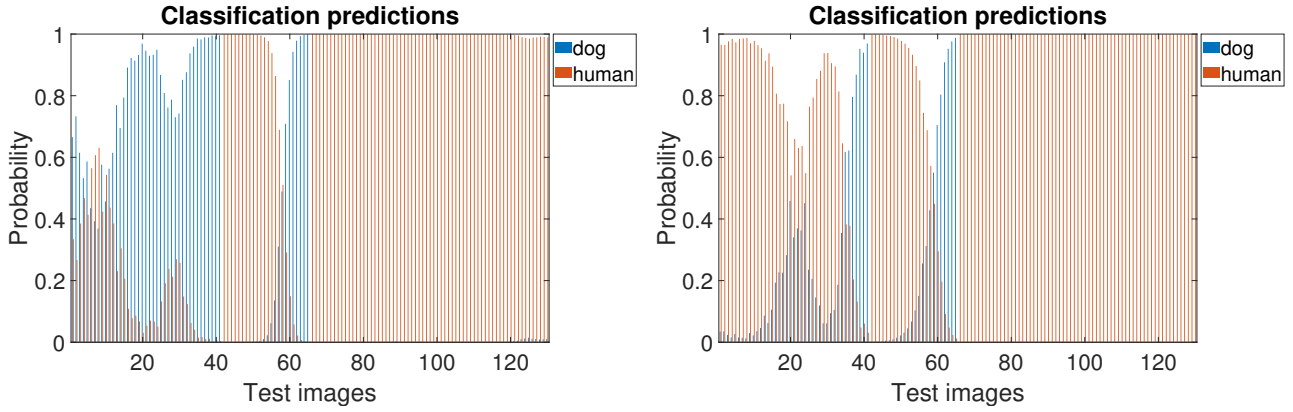
(f) $p = 0.99$

Figure 5.4: Training and validation loss curves from dropout test runs.



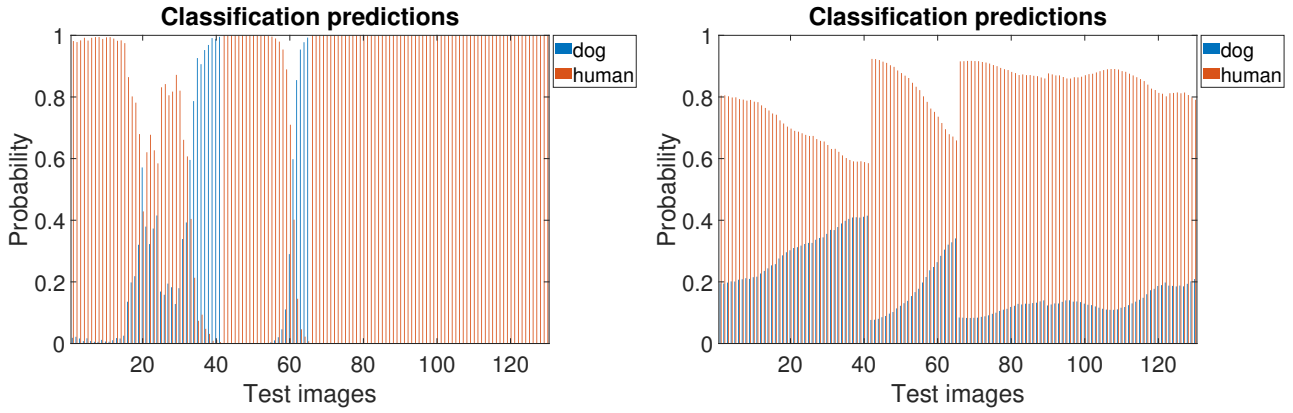
(a) $p = 0.1$

(b) $p = 0.2$



(c) $p = 0.3$

(d) $p = 0.4$



(e) $p = 0.5$

(f) $p = 0.99$

Figure 5.5: Predictions on simulated Im256 images from the dropout test runs. The 65 bars to the left are dog predictions and the 65 bars to the right are human predictions.

5.2.3 Dropout and regularization

In the earlier sections the isolated effect of L2-regularization and dropout has been studied. This can be seen as searching for an optimal solution in a one dimensional plane. Now, both the dropout and regularization techniques are tested together to look at the combined effect, which requires searching for a solution in a two dimensional plane. This involves looking for completely new optimal values which don't necessarily agree with the one dimensional solutions. To study this, a few regularization strengths and dropout probabilities were cross tested based on the results from testing L2-regularization and dropout individually.

The classification accuracies on the simulated and real data resulting from cross testing $p = 0.2$, $p = 0.3$, $\lambda = 0.01$ and $\lambda = 0.001$ are listed in table (5.3). The corresponding loss curves and bar plots with test predictions are shown in figure (5.6) and figure (5.7) respectively.

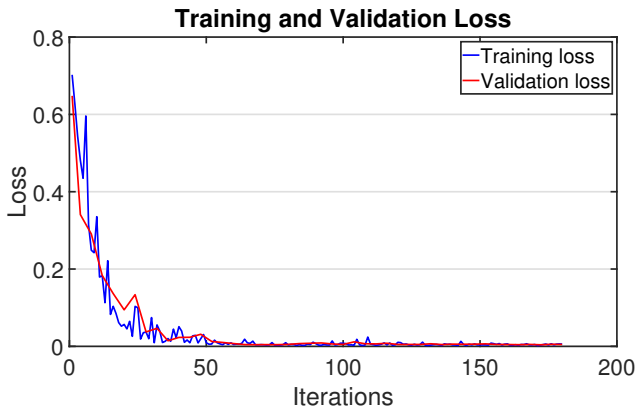
Im256			
p	λ	**Simulated data accuracy**	**Real data accuracy**
0.2	0.01	69%	50%
0.2	0.001	62%	45%
0.3	0.01	78%	40%
0.3	0.001	56%	43%

Table 5.3: Classification accuracies given dropout probabilities and regularization strengths when training on Im256 images.

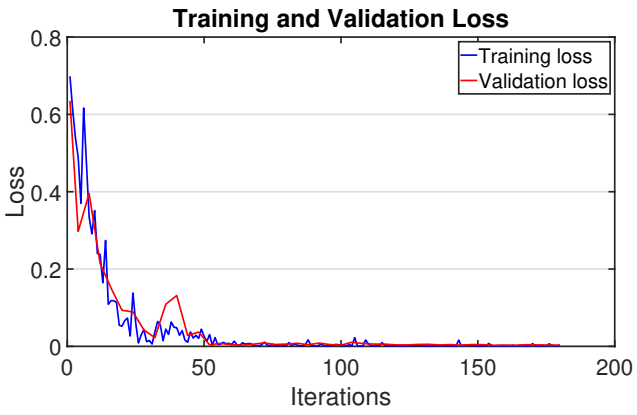
None of the selected parameter combinations resulted in improving the network's ability to classify the test data. Combining $p = 0.2$ and $\lambda = 0.01$ produced a classification accuracy of 69% on the simulated data, whereas $p = 0.2$ and $\lambda = 0.001$ achieved 62%. Their loss curves in figure (5.6a) and figure (5.6b) all converge to zero, however the predictions in figure (5.7a) and figure (5.7b) show poor performance on dog images, with the former predicting the dog images a little better.

The combination of $p = 0.3$ and $\lambda = 0.01$ resulted in the strongest classification accuracy of 78% on the simulated data. It's loss curves in figure (5.6c) decays nicely towards zero and the network is able to predict all human images with high confidence, whereas the dog images are predicted with less certainty, as seen in figure (5.7c). The last combination of $p = 0.3$ and $\lambda = 0.001$ scored a classification accuracy of 56%, only slightly better than chance. There is no apparent indication of this poor performance in the belonging loss curves in figure (5.6d), however the network predicts close to all images as human, as shown in figure

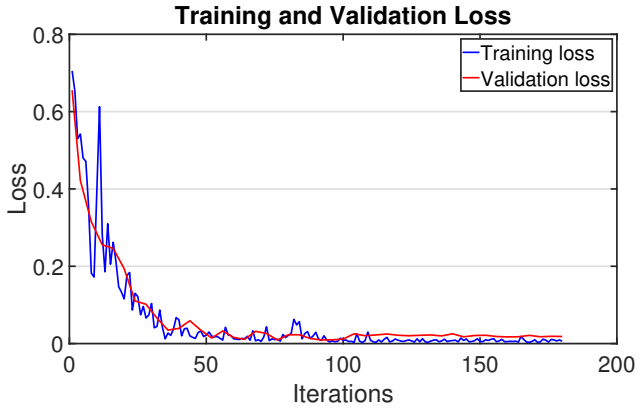
(5.7d).



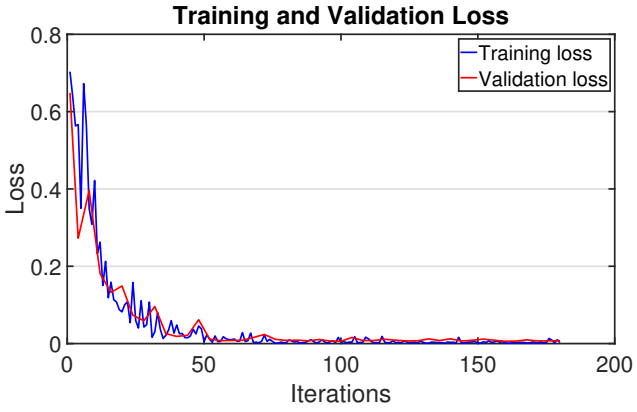
(a) $\lambda = 0.01$ and $p = 0.2$



(b) $\lambda = 0.001$ and $p = 0.2$

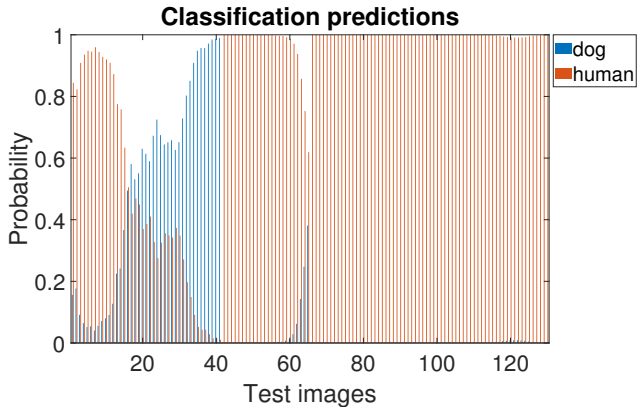


(c) $\lambda = 0.01$ and $p = 0.3$

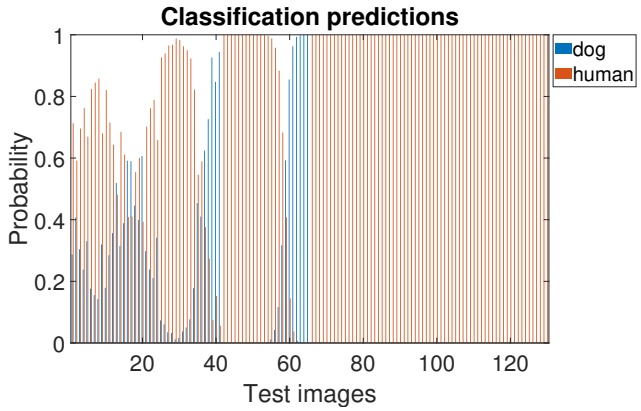


(d) $\lambda = 0.001$ and $p = 0.3$

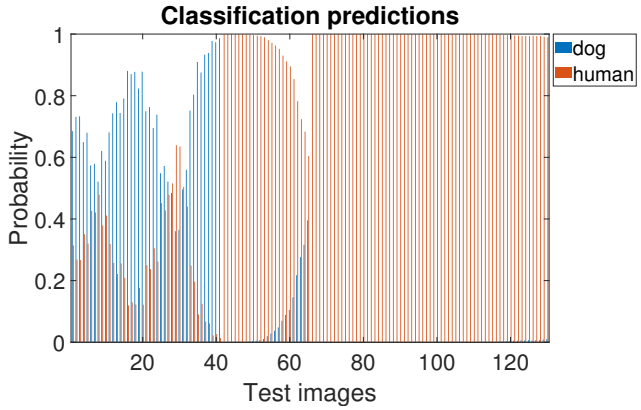
Figure 5.6: Training and validation loss on simulated test images given dropout probability p and regularization strength λ.



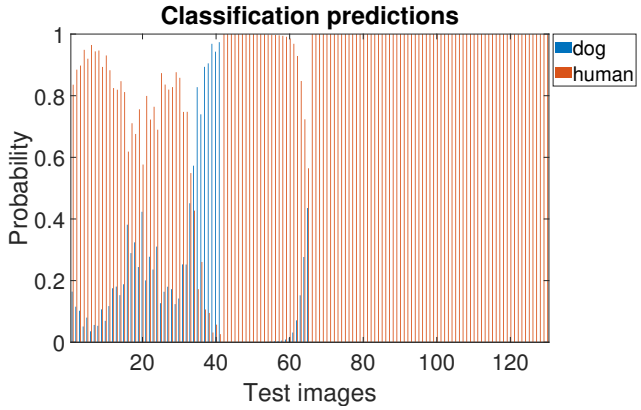
(a) $\lambda = 0.01$ and $p = 0.2$



(b) $\lambda = 0.001$ and $p = 0.2$



(c) $\lambda = 0.01$ and $p = 0.3$



(d) $\lambda = 0.001$ and $p = 0.3$

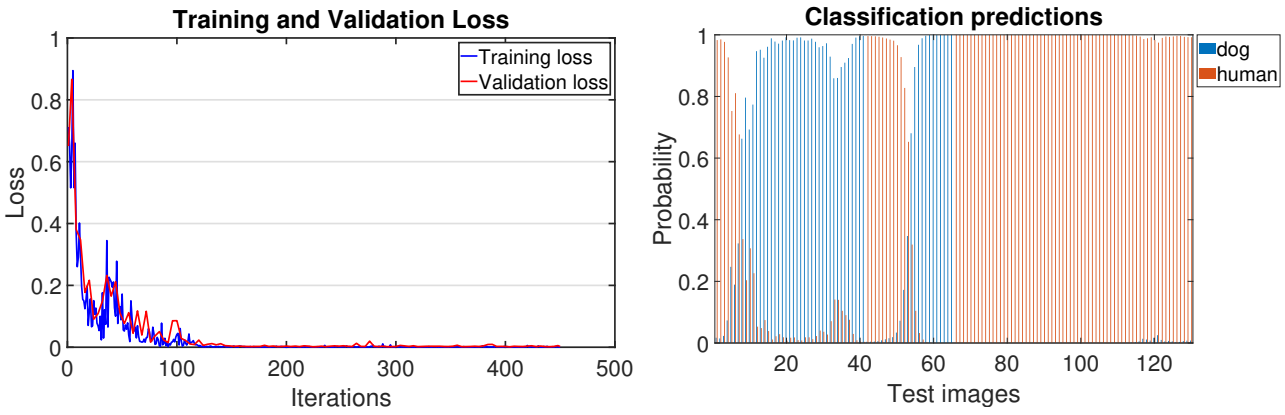
Figure 5.7: Predictions on simulated Im256 images given dropout probability p and regularization strength λ. The 65 bars to the left are dog predictions and the 65 bars to the right are human predictions.

These results merely represent the tip of the iceberg when exploring the various parameter combinations. They do however support the theory that the one dimensional results are not necessarily directly transferable to the two dimensional plane.

5.3 Testing on Im32 images

Similar tests are run on the Im32 images. Figure (5.8a) shows the loss curves when using no form of regularization or dropout. It reaches zero loss after about hundred and fifty iterations. This is a little slower than the case with the Im256 images. 85% accuracy was achieved on the simulated images, whereas 57% accuracy was achieved on the real images. The bar plot in figure (5.8b) shows strong predictions on all human images as well as the majority of the dog images. In the following sections, L2-regularization and the dropout technique are tested

on the Im32 images in an effort to improve performance.



(a) (b)

Figure 5.8: Loss curves and classification prediction on Im32 images.

5.3.1 Effect of regularization

The classification accuracies achieved when testing the Im32 images with different regularization strengths are listed in table (5.4). Corresponding training and validation loss curves are displayed in figure (5.9). The simulated test image predictions at the various regularization strengths are shown in figure (5.10).

Im32		
λ	**Simulated data accuracy**	**Real data accuracy**
1.0	50%	50%
0.1	94%	57%
0.01	88%	56%
0.001	86%	56%
0.0001	87%	56%

Table 5.4: Classification accuracies given regularization strengths tested on Im32 images.

The highest accuracy on the simulated data is 94% and is achieved with a regularization strength $\lambda = 0.1$. However, looking at the belonging loss curves in figure (5.9b) reveals losses not converging to zero. The bar plot in figure (5.10b) shows that the network classifies the test images with poor prediction levels, most around a probability of 0.5.

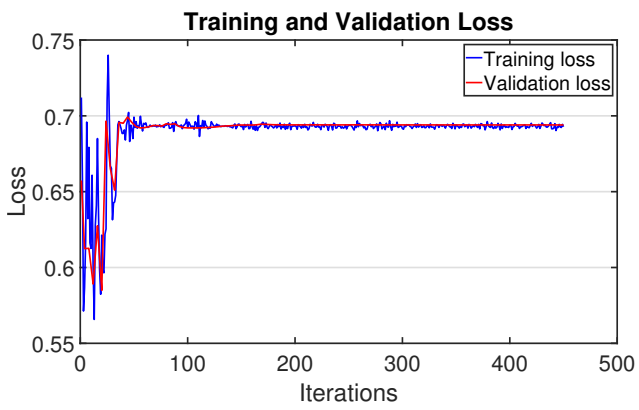
The second highest classification accuracy on the simulated data is 88%. It is achieved at $\lambda = 0.01$. The loss curve in figure (5.9c) converges to zero after approximately two hundred iterations and the bar plot in figure (5.10c) shows predictions with most probabilities

close to 1.

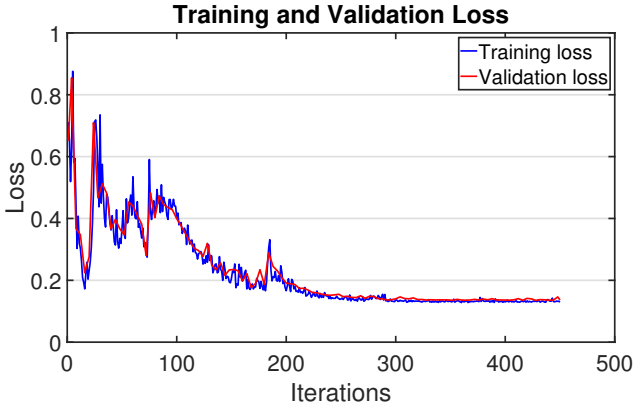
Further decreasing the regularization strengths leads to slightly lower classification accuracies, however the loss curves of figure (5.9d) and figure (5.9d) converges towards zero, also with strong levels of predictions as seen in figure(5.10d) and figure (5.10e).

The real data is classified poorly no matter what regularization strength is used. The highest classification accuracy achieved is 56.6% with $\lambda = 0.1$. It is however an improvement compared to the accuracies achieved when training on the Im256 dataset.

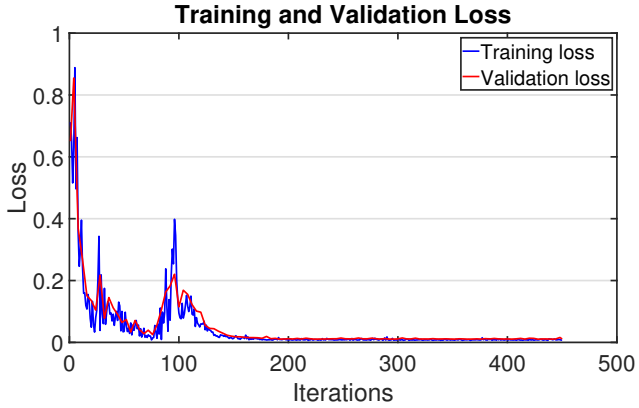
Using $\lambda = 1.$ results in 50% classification accuracy on both the simulated and the real data. The loss curve in figure (5.9a) stabilizes around 0.69 and the barplot in figure (5.10a) shows predictions at probabilities around 0.5. This is the same case as when training the Im256 images with $\lambda = 1$.



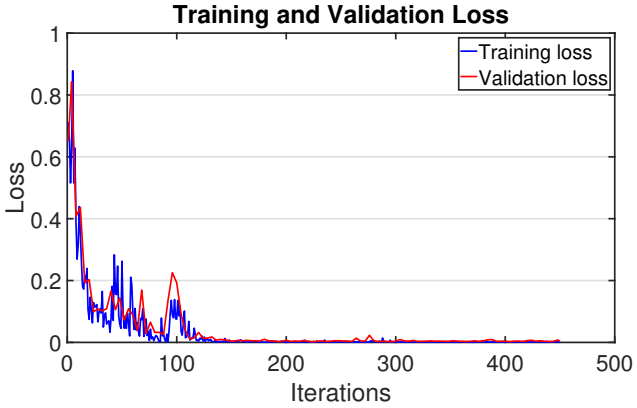
(a) $\lambda = 1$



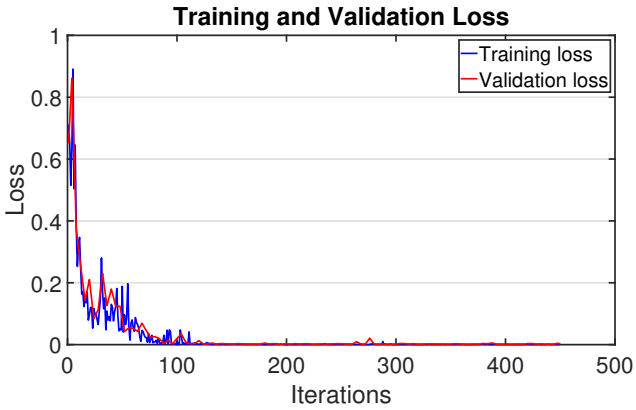
(b) $\lambda = 0.1$



(c) $\lambda = 0.01$

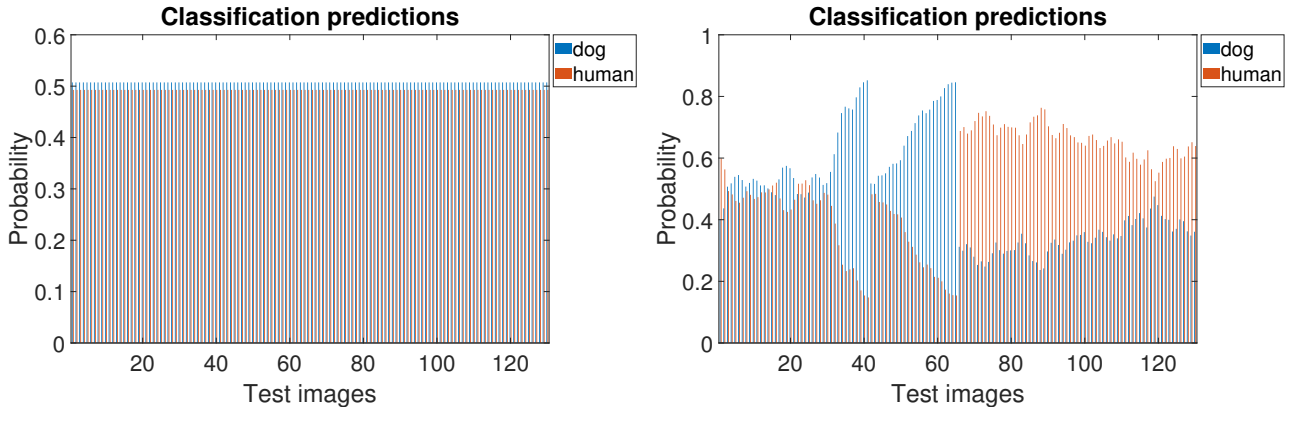


(d) $\lambda = 0.001$



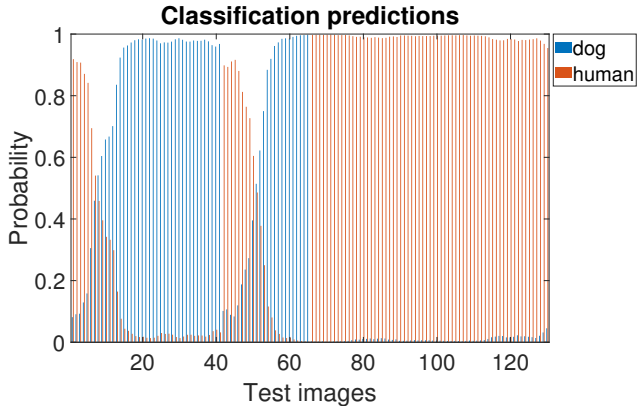
(e) $\lambda = 0.0001$

Figure 5.9: Training and validation loss curves from regularization test runs.

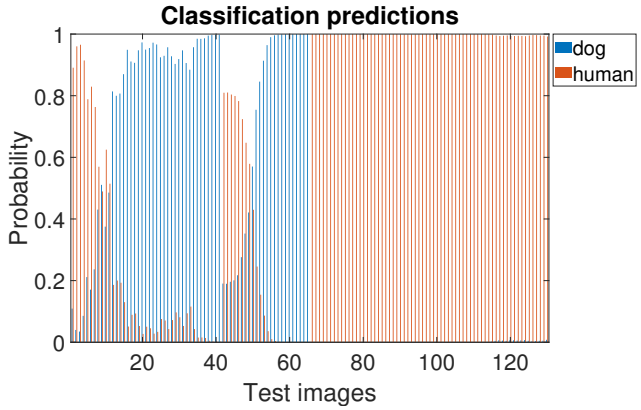


(a) $\lambda = 1$

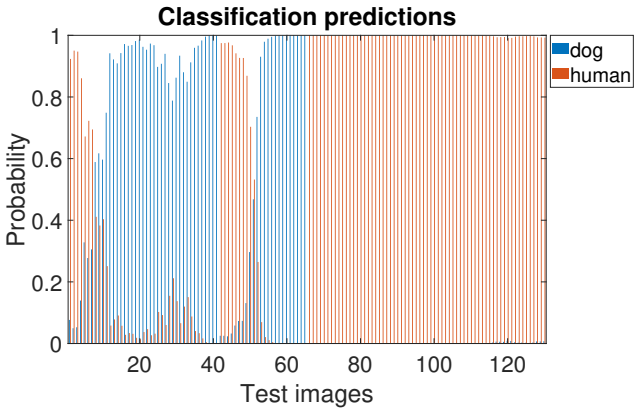
(b) $\lambda = 0.1$



(c) $\lambda = 0.01$



(d) $\lambda = 0.001$



(e) $\lambda = 0.0001$

Figure 5.10: Predictions on simulated Im32 test images from regularization test runs. The 65 bars to the left are dog predictions and the 65 bars to the right are human predictions.

5.3.2 Effect of dropout

Table (5.5) shows the results from testing the Im32 images with a dropout layer of different dropout probabilities. Figure (5.11) shows the corresponding loss curves and figure (5.12) shows the network's predictions when testing on the simulated data.

Im32		
p	**Simulated data accuracy**	**Real data accuracy**
0.1	100%	56%
0.2	79%	55%
0.3	100%	50%
0.4	99%	46%
0.5	95%	48%
0.99	84%	60%

Table 5.5: Classification accuracies given dropout probabilities tested on Im32 images.

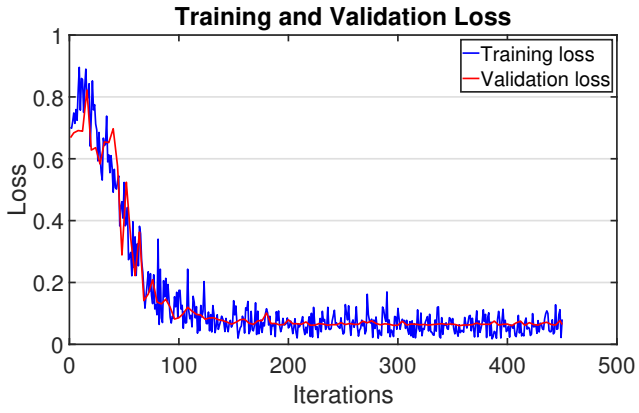
The simulated test images were 100% correctly classified using the probabilities of dropout $p = 0.1$ and $p = 0.3$. The belonging loss curves in figure (5.11f and 5.11d) converge towards zero. The bar plots in figure (5.12f and 5.12d) show strong predictions close to probabilities of 1. The latter, representing $p = 0.3$, shows the strongest network performance with all test images classified with high confidence.

Dropout probabilities $p = 0.4$ and $p = 0.5$ also scored high classification accuracies on the simulated test images with 99% and 95% respectively. These results are supported by nicely converging loss curves in figure (5.11c and 5.11b) and bar plots showing strong predictions in figure (5.12c and 5.12b).

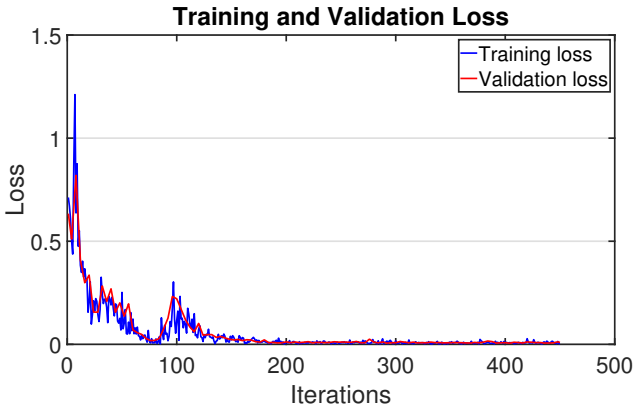
Lower classification accuracies on the simulated data were achieved with $p = 0.2$. The loss curves in figure (5.11e) show convergence towards zero, however the prediction bar plot in figure (5.12e) shows a portion of the dog images being incorrectly classified as human images.

The corner case with a probability of dropout $p = 0.99$ achieved a classification accuracy of 84% on the simulated data and 60% on the real data. The loss curves in figure (5.11a) stabilize around 0.1 loss and the bar plot shows predictions on simulated test data similar to that which was achieved using $p = 0.2$. When further increasing the probability of dropout the classification performance aggravates and produces classification accuracies around 50%. This suggests $p = 0.99$ is close to the dropout probability limit when training on the Im32 images.

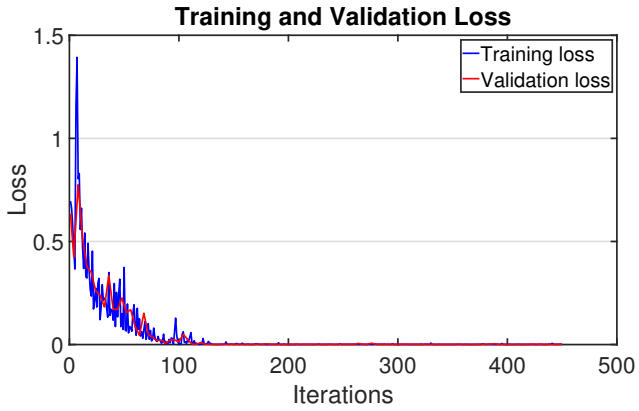
The real data is poorly classified with the various dropout probabilities tested with.



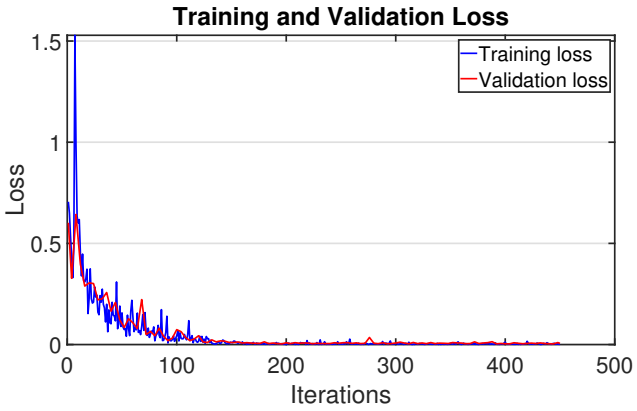
(a) $p = 0.99$



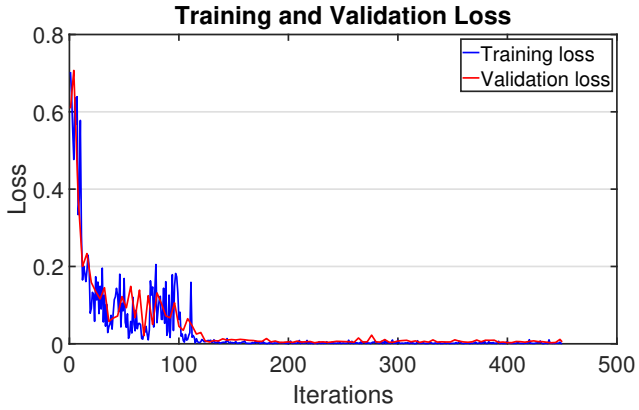
(b) $p = 0.5$



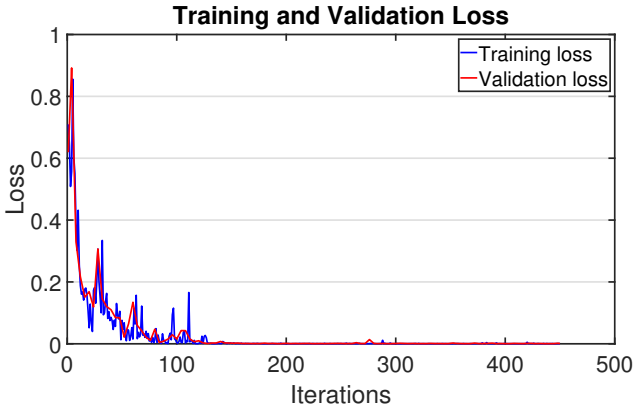
(c) $p = 0.4$



(d) $p = 0.3$

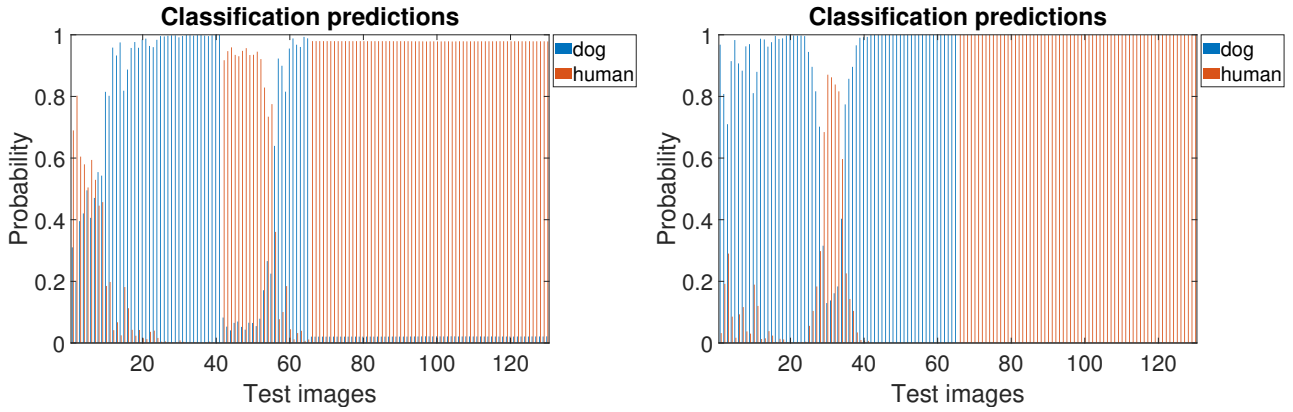


(e) $p = 0.2$



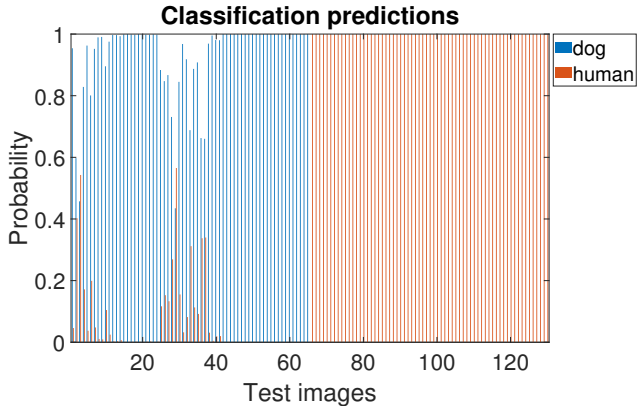
(f) $p = 0.1$

Figure 5.11: Training and validation loss curves from dropout test runs on Im32 images.

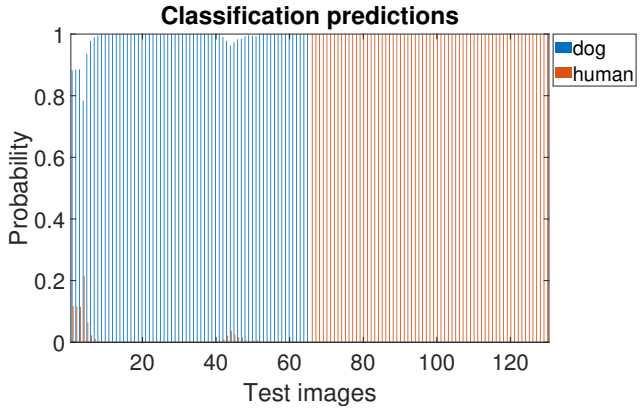


(a) $p = 0.99$

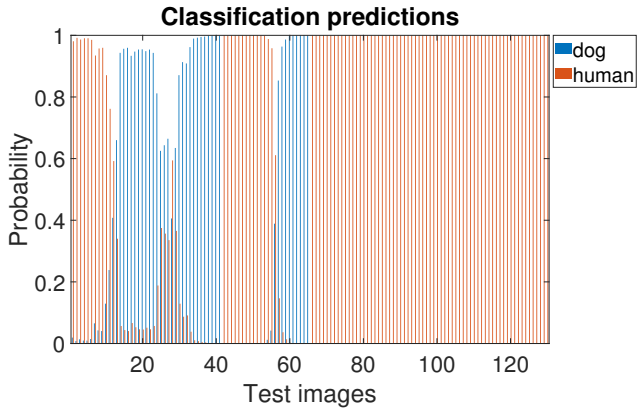
(b) $p = 0.5$



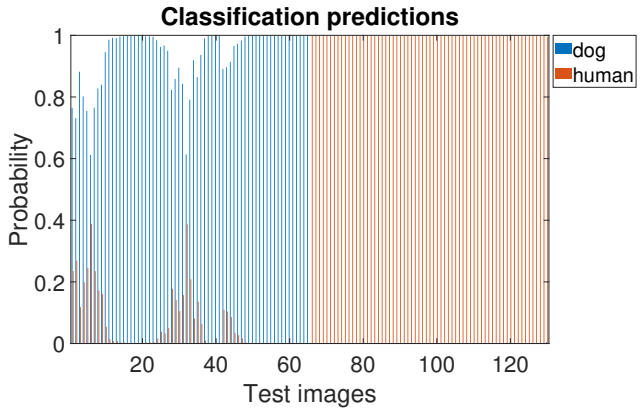
(c) $p = 0.4$



(d) $p = 0.3$



(e) $p = 0.2$



(f) $p = 0.1$

Figure 5.12: Predictions on simulated Im32 images from dropout test runs. The 65 bars to the left are dog predictions and the 65 bars to the right are human predictions.

5.3.3 Dropout and regularization

Again, the combined effect of dropout and L2-regularization is studied. The resulting classification accuracies on the simulated and real data are listed in table (5.6). The corresponding loss curves and bar plots with test predictions are shown in figure (5.13) and figure (5.14) respectively.

Im32			
p	λ	**Simulated data accuracy**	**Real data accuracy**
0.2	0.01	86%	58%
0.2	0.001	100%	51%
0.3	0.01	89%	58%
0.3	0.001	99%	56%

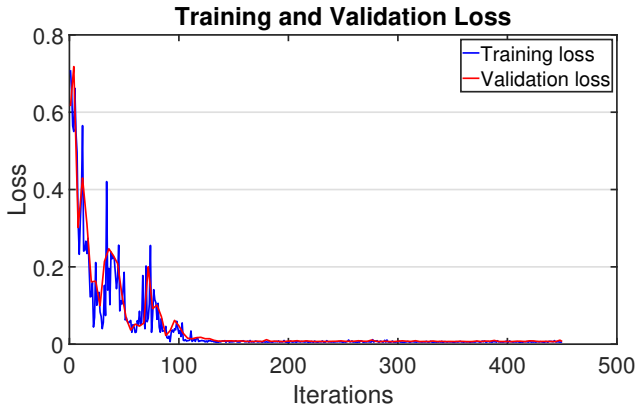
Table 5.6: Classification accuracies given dropout probabilities and regularization strengths when training on Im32 images.

The parameter combinations resulted in good classification accuracies overall. Combining $p = 0.2$ and $\lambda = 0.01$ produced a classification accuracy of 86% on the simulated data and 58% on the real data. Some distinctive oscillations occur in the loss curves of figure (5.13a) as they decay towards zero. The bars in figure (5.14a) express good predictions on both dog and human, however some dog images are predicted as human.

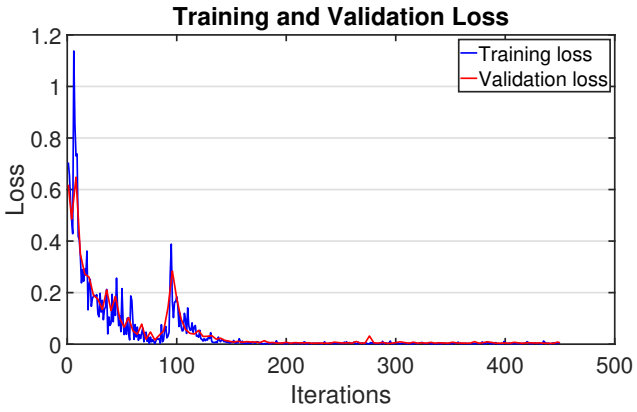
$p = 0.2$ and $\lambda = 0.001$ achieved 100% accuracy on the simulated images and 51% on the real images. The loss curves in figure (5.6b) converge to zero and the predictions in figure (5.7b) are very strong on both dog and human images with probabilities close to 1 throughout the test set.

Combining $p = 0.3$ and $\lambda = 0.01$ scored a classification accuracy of 89% and 58% on the real images. Also here, the corresponding loss curves show some oscillations before reaching zero loss, as seen in figure (5.6c). Figure (5.7c) shows that the network predicts human images with very high confidence, whereas the dog images are less easily labeled.

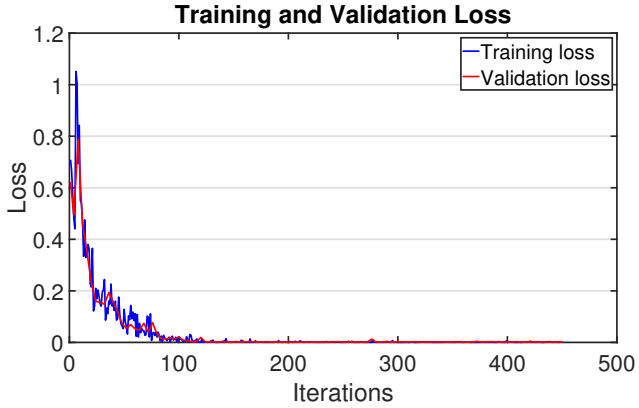
The last combination of $p = 0.3$ and $\lambda = 0.001$ resulted in 99% accuracy on the simulated data and 56% on the real data. Its loss curves in figure (5.6d) decay nicely towards zero and the network is able to predict all human images with high confidence. Figure (5.7d) shows that all human images are well predicted. The network has some difficulties predicting the dog images with high certainties.



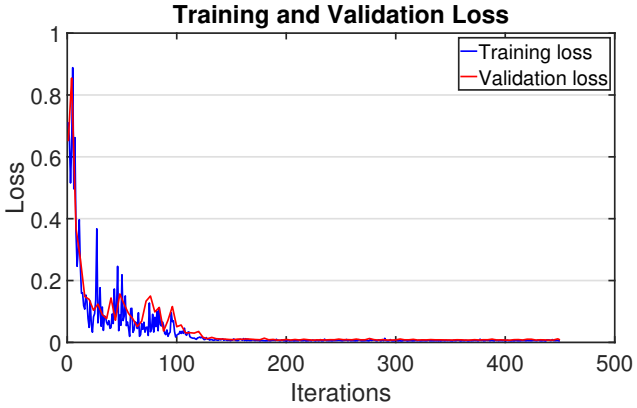
(a) $\lambda = 0.01$ and $p = 0.2$



(b) $\lambda = 0.001$ and $p = 0.2$

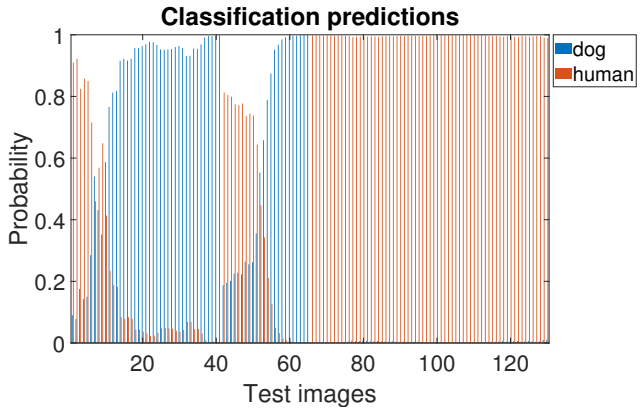


(c) $\lambda = 0.01$ and $p = 0.3$

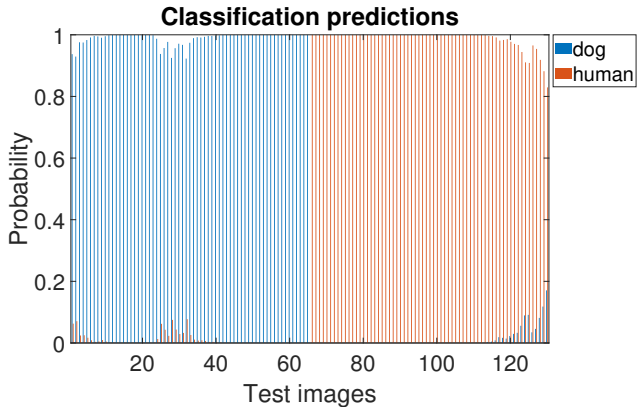


(d) $\lambda = 0.001$ and $p = 0.3$

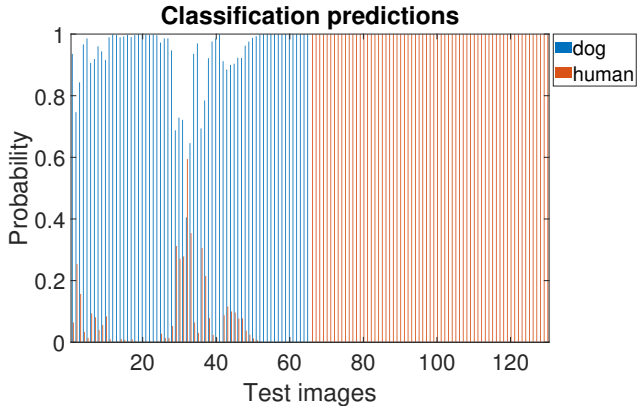
Figure 5.13: Training and validation loss on simulated test images given dropout probability p and regularization strength λ.



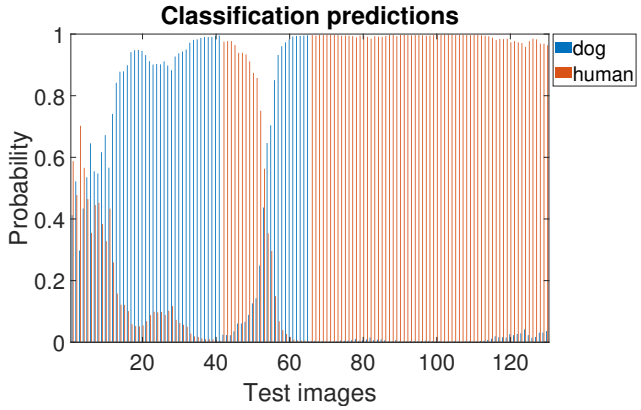
(a) $\lambda = 0.01$ and $p = 0.2$



(b) $\lambda = 0.001$ and $p = 0.2$



(c) $\lambda = 0.01$ and $p = 0.3$



(d) $\lambda = 0.001$ and $p = 0.3$

Figure 5.14: Predictions on simulated Im32 images given dropout probability p and regularization strength λ. The 65 bars to the left are dog predictions and the 65 bars to the right are human predictions.

Chapter 6

Discussion

6.1 Input images

6.1.1 Image design

The majority of the work during this thesis focused on preparing and testing different types of images. The final design as presented in chapter 4.2 was chosen as it was believed to display easily learnable information to the CNN. Perhaps the most important decisions are setting the window length and time length of the micro-Doppler signatures. A window length of 0.2 seconds in the STFT was found to display information in a smooth visual manner. A time length of one second was set in order to have the images contain frequency content over enough frames to capture distinct signatures. The PRF is dependent on the velocities of the targets of interest and must be set high enough to avoid aliasing.

Other configurations tested were images consisting of fewer frames, such as the image configurations studied in the earlier works of [11, 2]. Those configurations used a minimal amount of frames, generating images with only sudden changes of frequency visible. This proved to work poorly on the test sets. Further focus was therefore directed towards larger time series images. For the longer time series to be more suitable as inputs to the neural network they were downsampled into quadratic images.

6.1.2 Overlapping images

Every micro-Doppler image created overlapped the preceeding image with all but fifty frames. This effectively increased the size of the datasets. It can be argued that the overlapping images break with the independence criteria in the IID assumption, as presented in chapter 3.3.3. The part of the images that overlap has some content which is present in other images.

On the other hand, the constant shifting when creating new images results in a dataset where none of the images are exactly the same. With the amount of radar data available, this is perhaps the most efficient way of making a dataset of descent size.

6.1.3 Real data VS simulated data

Recalling from chapter 4.2.2 and the illustration in figure (4.5), clear oscillating frequency patterns are visible in the real images. The human and dog images can be distinguished by the frequency at which the patterns are oscillating, with the dog images swinging faster than the human images. The real images are however littered with noise in the form of clutter and propagation loss. This renders the patterns in a portion of the real dataset less recognizable. It also makes micro motions arising from swinging arms and legs less distinct. The simulated data on the other side show obvious micro motions around the bulk motion. The noise is non-existent and the signatures are very distinct, particularly in the human images. No form of propagation loss is added to the simulated data, which allows for uniform signal strength over time. The fact that the real and simulated images are so different may mean that basing classification off one other is a wasted effort. More resemblance between the images is likely necessary for good neural network predictions to become a reality.

6.2 Results

Two image types of different sizes were used as input data to the CNN architecture, namely the Im256 and Im32 images. The network was tested on a test set made up of simulated data as well as another test set comprised of real data acquired from Noveldas X4-radar.

6.2.1 Network initialization

Several efforts were made to enable the possibility to reproduce a neural network of well trained weights when training with identical parameters. Two weight initialization methods were tested. The default method in matlab is drawing numbers from a zero mean Gaussian distribution with unit variance. This resulted in highly varying weights and a large fluctuations in the results between identical training sessions. Recalling from chapter 3.4, the "HE-initialization" method is a robust method when training on ReLUs as the choice of activation functions. This method lead to less fluctuations, however the classification accuracies worsened. The focus was therefore redirected back to the default matlab method.

To produce the same series of random numbers, a seed was set in the main script. This sets a starting point in the random number generator when shuffling the input data and

initializing the weights of the neural network. The fluctuations decreased, although full reproducibility was still not achieved. This means the results presented in this work are not necessarily the optimal solutions from training and testing with the chosen network architecture and input data. They do however point out interesting network behaviours in various scenarios and may serve as guidelines in future image and network design decisions.

6.2.2 Testing on simulated data

The first round of testing on the CNN was done with the Im256 images. Without using any techniques to prevent overfitting, the network was able to predict 81% of the simulated images correctly. The incorrectly classified images belonged to the dog class. Performance improved when adding a regularization term to the cost function. 100% accuracy was achieved when using the regularization strength $\lambda = 0.1$, however the dog images were predicted with low probabilities, meaning the network made conclusions with low certainty. The belonging training and validation loss also indicated that this regularization strength is too high. Lowering the regularization strength to $\lambda = 0.01$ gave 89% classification accuracy with higher certainties in the predictions. The dropout technique contributed little when training the Im256 images. The highest classification accuracy achieved with dropout was 84% with a probability of dropout $p = 0.3$. Combining regularization and dropout did not improve classification in this case.

A breakthrough was made when testing the Im32 images. Using the dropout technique with the probabilities $p = 0.1$ and $p = 0.3$ during training resulted inn 100% classification accuracy on the test images, which were for the most part predicted with a probability of 1. This means the network classified the images with very high certainty. This was also achieved when using the dropout technique with $p = 0.2$ along with the regularization strength $\lambda = 0.001$. Using regularization alone didn't give results this good, however classification accuracies close to 90% were still achieved with various strenghts of regularization.

The increase in performance using the Im32 images could be a result of the final size of the image after being downsampled. Before being mapped onto the fully connected layer, the Im32 image is downsampled by five max pooling layers into 1×1 pixels. The Im256 images on the other hand are of size 8×8 after passing through the last max pooling layer. With this reasoning, it is possible that adding more max pooling layers, which further downsamples the images, would improve prediction when using the Im256 images.

Another explanation is that the high resolution of the Im256 images might be abundant. The major components of the micro-Doppler signatures are still visible to the human eye

in the downsampled Im32 images. If the CNN is able to capture these features and make predictions, the extra information gained by the high resolution in the Im256 images will only be a source of noise.

6.2.3 Predicting dogs as humans

A typical problem when testing on both image types was dog images being predicted as human. Human images in general were more easily predicted. This is likely a direct result of the higher access to diverse human data. To explore how the network predicts the test images, some of the simulated test images are displayed as they are seen by the CNN along with their predictions in figure (6.1).

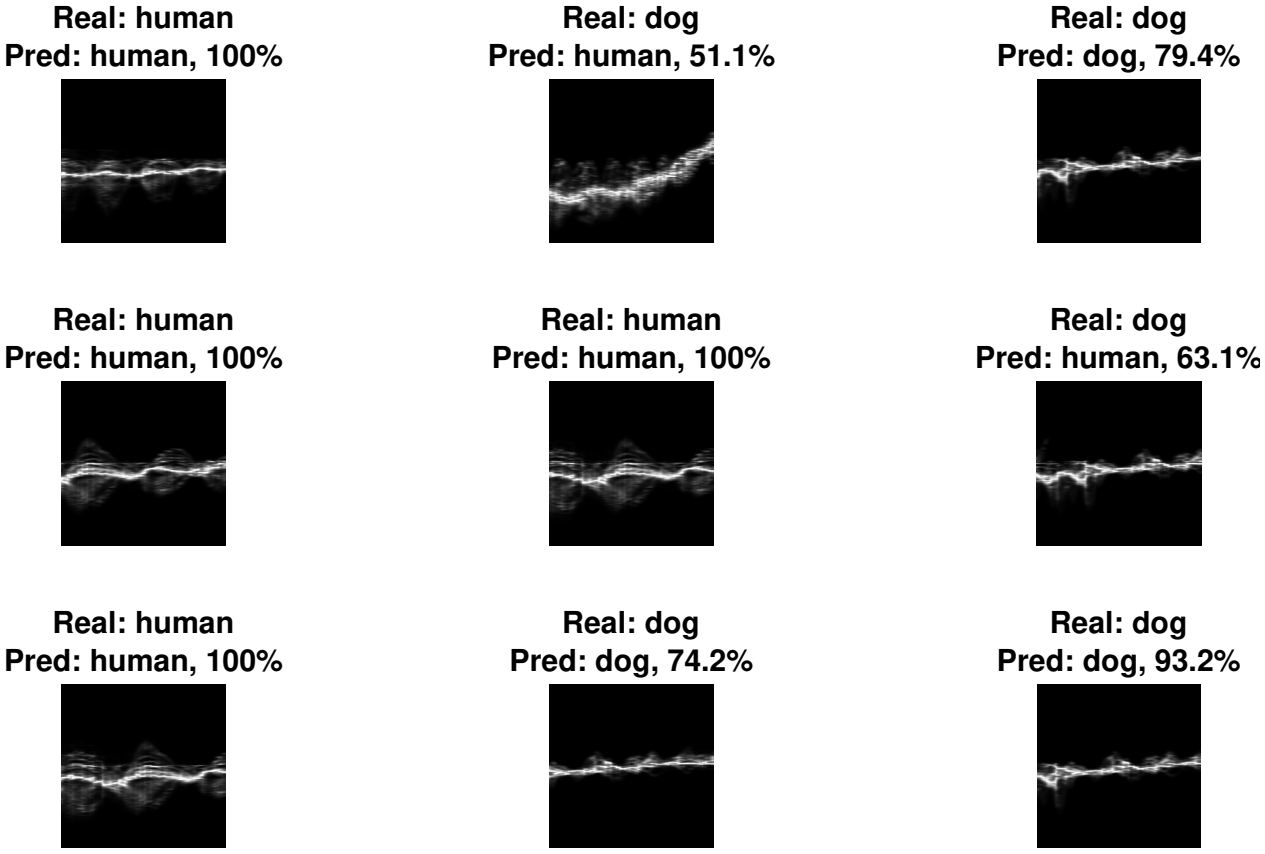


Figure 6.1: Images of type Im256 along with their network predictions.

This example is from training the Im256 images with a dropout probability $p = 0.3$, where the achieved accuracy was 84%. All human images are classified with 100% confidence. Distinct oscillations in frequency can be seen in the human images. The dog images are correctly classified with predictions between 74.2% and 93.2%. Dog images are distinguishable from human images in that the oscillations are weaker and closer to the bulk frequency which can

be seen as the brightest lines. Two dog images are incorrectly predicted as human: The top image in the middle separates from the other dog images in that it has more frequency content around its bulk motion frequency. This might be a feature that the network recognizes as human. The image in the middle to the right is similar to other dog images, however the image is still predicted as human. Telling how the network interprets the images is a far-fetched affair. Some intuition can be gained by studying the filters in the convolutional layers, some of which are added to appendix A.

6.3 Future Work

6.3.1 Simulate data based on mathematical models

The use of the simulator was limited to motion capture files on humans and dogs, where the access to human data was much greater than that of dogs. The need for more data on dogs is crucial to be able to create a good neural network classifier. For simulated data this can be done by either making more motion capture files, or by making a target generator based on mathematical models, which has been studied in [6]. The first choice allows for highly realistic generating of data. The latter option opens up for a large variety of scenarios where the mathematical models can be altered to cover different gait velocities, target sizes and radial angles between radar and target. This is perhaps the most useful option if appropriate mathematical models can be made and could therefore be a field of interest in future work.

6.3.2 Work with real data

The neural network architecture tested in this thesis was solely trained on simulated data as the quantity and quality of the simulated data was superior to that of the real data. The resulting network was able to classify simulated data with up to 100% accuracy. Prediction attempts were done on a test set comprised of real data to see if it could be classified based on simulated data, however the real data proved to be too different from the simulated data. Improving the quality of the real data is believed necessary in order to extract micro-Doppler signatures that can be classified with the same level of confidence as the simulated data. To improve the quality when acquiring more data, two key points need to be considered:

- Increase the PRF to avoid folding and capture high velocity targets.
- Obtain a continuous range track over time.

During the second round of data acquisition the winter of 2018, much of the data were rendered useless due to the targets moving at velocities exceeding the limit given by the PRF

being used. This violated Nyquist's sampling criterion causing the data to fold, as described in chapter 2.2.1. This illustrates the importance of using a high PRF when capturing radar data.

In order to acquire micro-Doppler signatures with a minimal amount of noise it is necessary to obtain the range to the target over time. During the first round of recordings in the fall of 2017 this was acquired by the use of a range sensor placed on the target of interest. Another solution is to track major components in the range Doppler matrix. A tracking algorithm has been implemented by Novelda to handle this. It groups moving content into clusters and attempts to track each cluster. Such a cluster can be a moving arm. It has proved difficult to maintain track over a longer period of time. This limitation makes the second round of recordings harder to use, as extracting micro-Doppler images requires a constant track over all the relevant frames. It does however have to potential of obtaining range tracks that relate to all moving body parts which can help generate clean micro-Doppler signatures.

The network was able to pick up features good enough to discriminate between simulated dog and human images. It should also be able to classify real data if real data of good quality is included when training the CNN. For this reason, finding good radar settings and acquiring more real data using Novelda's X4-radar should be given priority.

6.3.3 Explore other network configurations

The field of deep learning is developing fast and there are many ways to design a network. The CNN design in this thesis achieved good classification results on simulated data, by exploring the effects of L2-regularization and dropout. Different learning rates were tested as well with the final setup being a drop schedule. This gradually decreased the learning rate, allowing for a rough search for the optimal solution in the beginning followed by smaller and smaller step sizes in the SGD method as the epochs finished. However, a great deal of methods and parameters have not yet been tested that could improve learning. Some suggestions to what can be studied further are

- Weight initialization.
- Choice of gradient descent method.
- Transfer learning.

Bibliography

[1] Xethru homepage, 2018.

[2] Harald Blehr. *ANN based respiration detection using UWB radar*. NTNU Department of Engineering Cybernetics, 2017.

[3] Victor C. Chen. *The Micro-Doppler Effect in Radar*. Artech House, 2011.

[4] X. Glorot, A Bordes, and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks, pages 249-256, 2010.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, Spring 2017.

[6] Helge Langen. *Ultra-Wideband Radar Simulator for classifying Humans and Animals based on Micro-Doppler Signatures*. NTNU Department of Electronics and Telecommunications, June 2016.

[7] Anders Liland. *Development of a pulse-Doppler radar simulator and micro-Doppler feature extractor used for Automatic Target Recognition*. NTNU Department of Electronics Systems Design and Innovation, 2018.

[8] Michael Nielsen. *Neural Networks and Deep Learning*. 2017.

[9] Mark A. Richards, James A. Scheer, and William A. Holm. *Princiles of Modern Radar Vol. I: Basic Principles*. Scitech Publishing, 2010.

[10] Stanford University. Cs231n convolutional neural networks for visual recognition, Spring 2017.

[11] Børge Wiik. Ann based classification of humans and domestic animals using pulse doppler radar. 2017.

[12] Thor Øyvind Fossum. *Exploration of Micro-Doppler Signatures Associated with Humans and Dogs using UWB radar*. NTNU Department of Electronics and Telecommunications, 2015.

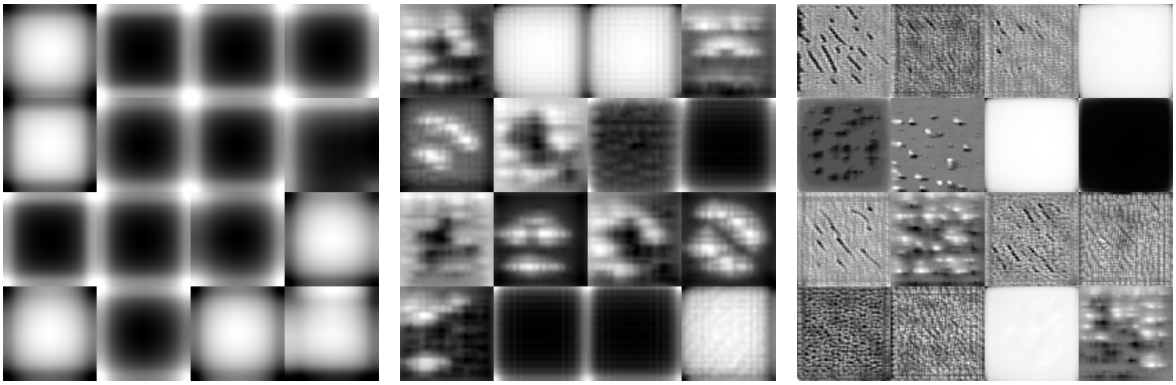
Appendix A

Filters

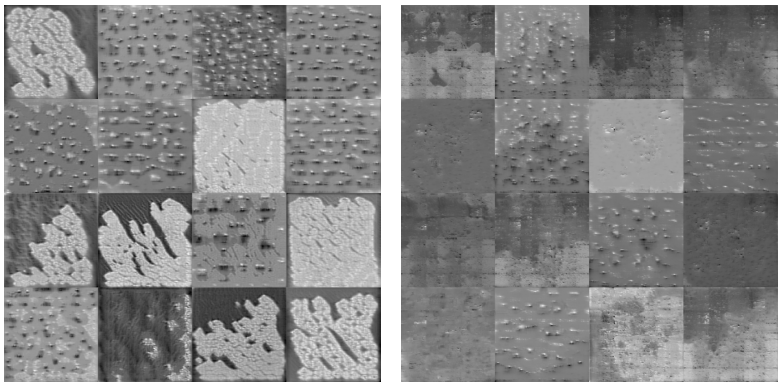
The filters of the convolutional layers display interesting information. The filters from two training configurations with strong results are presented here. 16 filters are drawn from each convolutional layer and displayed.

Figure (A.1) shows filters resulting from training the Im256 images with regularization strength $\lambda = 0.01$. Many of the filters in the first convolutional layer look similar. Some edges can be eyeballed. For the proceeding layers the details get more specific, with blobs and edges clearly visible in the second layer. The patterns in the three last layers get more abstract. Some of the filters seem to display no information at all.

Similar cases arise in the filters resulting from training the Im32 images. Figure (A.2) displays filters from training with a regularization $\lambda = 0.001$ and dropout probability $p = 0.2$. This configuration seems to have a larger portion of the filters displaying visible patterns than the filters in figure (A.1). This may indicate that features are more easily learned when training on smaller sized images.

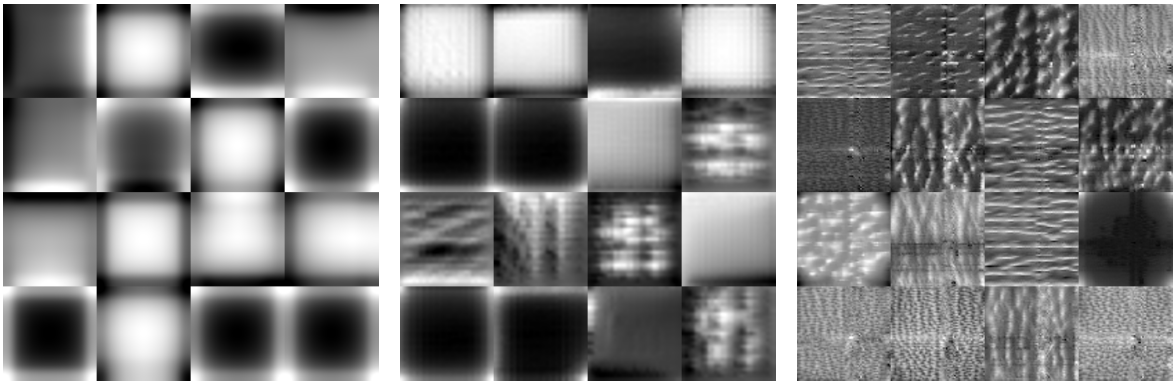


(a) 1st convolutional layer (b) 2nd convolutional layer (c) 3rd convolutional layer

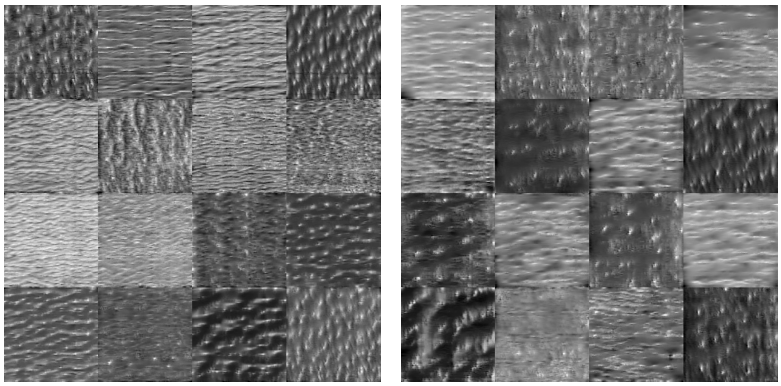


(d) 4th convolutional layer (e) 5th convolutional layer

Figure A.1: Filters of convolutional layers when using regularization with the Im256 images.



(a) 1st convolutional layer (b) 2nd convolutional layer (c) 3rd convolutional layer



(d) 4th convolutional layer (e) 5th convolutional layer

Figure A.2: Filters of convolutional layers when using regularization and dropout with the Im32 images.