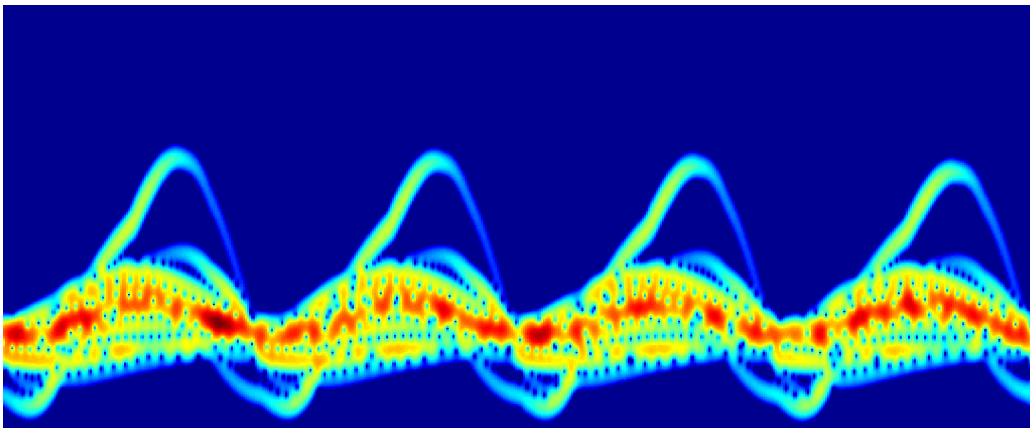
An Analysis of Deep Learning for Human Gait Classification in Radar



Roeland Trommel

An Analysis of Deep Learning for Human Gait Classification in Radar

Roeland Trommel

Approval Internship report/Thesis of:

Roeland Trommel

Title: An Analysis of Deep Learning for Human Gait Classification in Radar

Educational institution: Delft University of Technology

Internship/Graduation period: 1 Sept 2015 - 22 Jun 2016

Location/Department: Delft, Advanced Development

Thales Supervisor: R. I. A. Harmanny

This report (both the paper and electronic version) has been read and commented on by the supervisor of Thales Netherlands B.V. In doing so, the supervisor has reviewed the contents and considering their sensitivity, also information included therein such as floor plans, technical specifications, commercial confidential information and organizational charts that contain names. Based on this, the supervisor has decided the following:

- ○ This report is **publicly available (Open).** Any defence may take place publicly and the report may be included in public libraries and/or published in knowledge bases.

- ☑ This report and/or a summary thereof is **publicly available to a limited extent (Thales Group Internal)**.
  It will be read and reviewed exclusively by teachers and if necessary by members of the examination board or review committee. The content will be kept confidential and not disseminated through publication or inclusion in public libraries and/or knowledge bases. Digital files are deleted from personal IT resources immediately following graduation, unless the student has obtained explicit permission to keep these files (in part or in full). Any defence of the thesis may take place **in public to a limited extent**. Only relatives to the first degree and teachers of the N/A department <*name department* > may be present at the defence.

- ○ This report and/or a summary thereof, is **not publicly available (Thales Group Confidential)**. It will be reviewed and assessed exclusively by the supervisors within the university/college, possibly by a second reviewer and if necessary by members of the examination board or review committee. The contents shall be kept confidential and not disseminated in any manner whatsoever. The report shall not be published or included in public libraries and/or published in knowledge bases. Digital files shall be deleted from personal IT resources immediately following graduation. Any defence of the thesis must take place **in a closed session** that is, only in the presence of the intern, supervisor(s) and assessors. Where appropriate, an adapted version of report must be prepared for the educational institution.

Approved:

Harmanny

(Thales Supervisor)

Approved:

(Educational institution)

Delft, 9 June 2016

(city/date)

(copy security)

An Analysis of Deep Learning for Human Gait Classification in Radar

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

by

Roeland Trommel
born in IJsselmuiden, the Netherlands

Microwave Sensing, Signals & Systems
Department of Microelectronics
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
radar.ewi.tudelft.nl

Thales Nederland BV
Advanced Development, Delft
Delft, the Netherlands

An Analysis of Deep Learning for Human Gait Classification in Radar

Author: Roeland Trommel
Student id: 4000986
Email: rptrommel@gmail.com

Abstract

Over the past five years, deep learning techniques have led to astounding breakthroughs in many areas, like speech recognition and computer vision applications. The application of deep learning in the radar domain has however been rather limited. The merit of using deep learning techniques for the purpose of human gait classification in radar is investigated. Several models based on three deep neural network architectures, the multi-layer perceptron, the autoencoder and the convolutional neural network, are used to distinguish human walking and running gaits from non-gait signatures in radar micro-Doppler spectrograms. The effects of model architecture, size and depth are analyzed and convolutional models are proven to be the most effective. A deep convolutional neural network (DCNN) is designed to distinguish the number of human gaits in a multi-target classification scenario. Experimental data includes synthetic data at several radar frequencies and SNR levels, as well as X-band CW radar measurements taken at various ranges. The effects on the performance of the DCNN are investigated by varying with regularization methods, the amount of training data, training algorithms, parameter initialization, and transfer learning. Deep learning techniques, and convolutional neural networks in particular, are proven to be an effective approach for human gait classification in radar.

Thesis Committee:

Chair:	Prof. DSc. A. Yarovoy, Faculty EEMCS, TU Delft
Supervisor:	Dr. J.N. Driessen, Faculty EEMCS, TU Delft
Committee Member:	Dr. D.M.J. Tax, Faculty EEMCS, TU Delft
Company supervisor:	Ir. R.I.A. Harmanny, Thales Nederland BV

Preface

One does not simply complete a Master Thesis and this one in particular would not have been possible without the help of many people. First of all, I would like to express my immense gratitude to my supervisors Ronny Harmanny and Lorenzo Cifola at Thales for their helpful comments, advice and making this graduation project a wonderful experience. Also, I would like to thank Hans Driessen for his comments and supervision. Furthermore, I would like to thank Alexander Yarovoy for chairing the thesis committee and his valuable advice. Also, I would like to thank thesis committee member David Tax for reading my thesis and his advice in the early stage of the research. And off course I would like to thank all the people who were so gracious to partake in my experiments and provide me with invaluable data. Finally I would like to thank my family and friends for their love and support during my study and graduation.

Roeland Trommel
Delft, the Netherlands
June 15, 2016

Contents

1

Introduction

Deep Learning is a subfield of machine learning based on computational models comprised of multiple abstraction levels, and it has brought many breakthroughs, greatly improving on the state-of-the-art in computer vision and speech recognition applications [1]. The application of Deep Learning in the radar domain has been however rather limited and it is interesting to investigate whether the Deep Learning paradigm can bring advances in the radar domain as well.

The goal of this project is to assess the merit of using Deep Learning techniques in the radar domain, more specifically for the purpose of radar target classification. Different approaches are possible, depending on the classification problem, the type of radar as well as the type of radar data that can be used for the classifier's input. For this research an a-priori trade-off was made between the different possibilities that would allow a useful evaluation of Deep Learning techniques, compared to older, say conventional, classification methods.

The chosen problem domain is that of human gait classification, based on radar micro-Doppler signatures as function of time, obtained by an X-band radar ($\sim$10 GHz). The human gait classification problem is of interest because of its importance in security & surveillance applications. This particular topic has also a few practical advantages with respect to other interesting targets: human gait can be easily measured with radar, and also models of human gait are available. Hence, acquiring either synthetic or experimental datasets of considerable size is feasible.

The main classification problem considered in this research project is to detect the presence of human gait(s) *and* to distinguish the number of human gait(s) using micro-Doppler signatures. This scenario has not been described in literature and due to the presence of possibly multiple targets, it forms a new and challenging test case that is well-suited for assessing the merit of Deep Learning based target classification in the radar domain.

In this thesis, a deep feedforward neural network is designed for the classification of human gaits. This is done using an incremental approach: first, a few possible network architectures are considered and the most suitable type of architecture is determined by small-scale experiments in which a walking human is distinguished from a running

human and a non-gait class. Next, a new model based on the selected architecture is designed and applied to the number of human gaits classification problem using model data to analyze it capabilities and properties, and the effects of various parts on the model's performance. This experience is used to improve the network model and finally, the updated model is applied to experimental data of human gaits for validation and analysis of the model's performance. A 3-NN classifier is used to compare the Deep Learning approach with a more conventional classifier.

This thesis is organized as follows: in chapter 2 an introduction to the field of Deep Learning and neural networks is provided, followed by a discussion of the basics of radar. The background chapter concludes with a discussion of human gait modeling and classification in radar, including a review of the state of the art. In chapter 3 the research questions to achieve the above described goal have been formulated, and the approach and materials used in this project are described. The results are presented in chapter 4, and in chapter 5 the findings and limitations of this study are discussed. Final conclusions and recommendations for future work are given in chapter 6.

2

Background

This chapter will provide introductory background material. An introduction to the field of deep learning is provided in section 2.1. Section 2.2 will discuss some fundamentals of radar including the Doppler and micro-Doppler effect. In section 2.3 human gait classification in radar will be discussed, including a description of human motion model and a review on the state of the art.

2.1 Deep Learning Fundamentals

2.1.1 Introduction

Machine learning is the field of study that gives computers the ability to learn from data without being explicitly programmed; a definition by Arthur Samuel in 1959. Deep Learning is a much more recent and broad term covering a set of techniques and algorithms in the field of machine learning that aim to construct multiple levels of representation or learning a hierarchy of features automatically from data, as defined in the forthcoming handbook of Goodfellow et al. [2]. Having multiple representations is valuable because some representations are more suitable for performing a task (e.g. classification, facial recognition etc.) than others and perhaps offer more insight in the problem at hand. In many applications, much effort is put into designing suitable representations in order to solve a particular problem. With Deep Learning, this stage of feature engineering is unnecessary as the model learns features on its own and hopefully develops an effective feature representation for the particular task at hand. In the fields of computer vision and speech recognition many astounding results have been achieved by models employing Deep Learning techniques, vastly improving on the state-of-the-art as LeCun, Bengio and Hinton point out in [1].

This section will discuss a number of machine learning basics and Deep Learning in particular. As the literature about machine learning and Deep Learning is vast and rapidly expanding, only a few main concepts can be treated. There are dozens of different types of models which can be considered as Deep Learning techniques. In this thesis, attention is restricted to the class of feedforward neural networks. This section will

introduce the main concepts of machine learning and Deep Learning and give an overview of techniques and building blocks that can be used to construct a deep neural network classifier. This section is organized as follows: first, in section 2.1.2 general machine learning and deep learning concepts will be discussed using the early and general type of artificial neural network called Multi-Layer Perceptron as a guiding example. Next, in section 2.1.3 two types of feedforward neural network models are discussed in more detail: the AutoEncoder (AE) and Convolutional Neural Network (CNN).

2.1.2 Basics of Deep Learning

Artificial neural networks

Deep Learning networks all have their roots in the many decades old artificial neural network which is loosely inspired by neuroscience. The artificial neural network (ANN) is composed of single units called nodes or neurons which are organized in layers and interconnected. A general feedforward ANN or Multi-Layer Perceptron (MLP) consists of an input layer, one or multiple so-called hidden layers and an output layer. Each layer has a number of neurons (or nodes) where each neuron is connected to all neurons of the previous layer and to all the neurons of the next layer, but there are no connections between neurons of the same layer (see figure 2.1). The output of a neuron is fed as input to all neurons of the next layer, and so on until the output layer is reached. Information travels only in the forward direction from the input layer towards the output layer, hence the name feedforward.

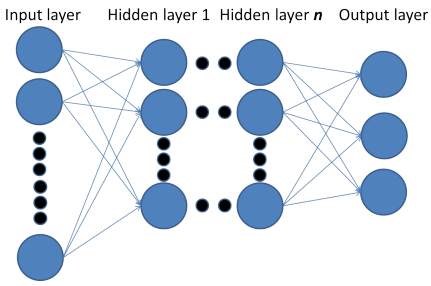


Figure 2.1: General Multi-Layer Perceptron (MLP) architecture.

Each neuron implements a mathematical function which is referred to as the activation function to provide a single output or activation based on its inputs.

$$f(x) = \sigma(\sum_{i=1}^{n} x_i w_i + b) \tag{2.1}$$

Here, x_i is the input i, w_i is the weight associated with input i, b is a bias term. It is by adapting these weights and biases of all the neurons that a neural network learns to provide the correct output for all kinds of inputs. The activation function σ can take

many forms and is generally non-linear. For a long time, the sigmoid function

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{2.2}$$

and hyperbolic tangent function

$$\tanh(x) = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} \tag{2.3}$$

were the default choices. Jarret et al. introduced the rectified linear unit or ReLU [3] and nowadays it has become the most popular activation function.

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.4}$$

It exhibits much better performance than the former sigmoid and hyperbolic tangent functions, especially in the sense that the network learns or converges faster as shown by Krizhevsky et al. [4].

Learning methods

Learning can be done in various ways and generally, three main different types of learning are recognized: supervised, unsupervised and reinforcement learning. In supervised learning, for each training example the machine learning system is provided with the correct output, e.g. a specific class label in the classification setting. The system can then adjust itself to better approximate the correct output. The machine must thus infer a function from the data to map the inputs to a desired output value and hopefully, it will learn a function that performs well on unseen inputs too. The unsupervised learning setting is about finding patterns in the data without receiving an error or reward signal to evaluate the solutions the machine finds. Important examples of unsupervised learning include density estimation techniques which try to find a statistical model underlying the input, and feature extraction which tries to extract statistical regularities directly from the inputs. In reinforcement learning, the machine interacts with its environment and it can influence which experience (training data) it will receive. There is no explicit correction for sub-optimal actions and usually there is a trade-off between exploration (trying new experiences) and exploitation (using known good experiences). In general, it is possible to combine different learning methods. Especially in the Deep Learning setting hybrid systems can be encountered. Usually this involves training a network or first stages of a network unsupervised and then only use supervised learning to fine-tune. The possibility of such a hybrid system is useful because acquiring unlabeled data is usually easy, whereas acquiring labeled data is often costly. In [5, 6] this hybrid approach was used to great effect. However, using unsupervised learning as pretraining seems to be only worthwhile when little labeled data is available. When large amounts of labeled data are available it has little benefit and it has generally fallen out of use in these cases [7].

Training of neural networks

Whether supervised or unsupervised, a neural network needs to be trained. Training a model means adjusting its parameters such that its performance on a task increases (e.g. achieving a lower classification error). In practice, training a model consists of minimizing a cost function which can be the ultimate goal of the network but often takes a different form which is easier to express and minimize, yet is (hopefully) strongly related to the ultimate goal. The two most common cost functions are the mean squared error and the categorical cross-entropy. The mean squared error is defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{2.5}$$

The sum is taken over the n data examples, $\hat{y}_i$ and y_i are the computed output and desired output respectively for a particular data point. The categorical cross-entropy loss is defined as

$$CE = -\sum_i \sum_j t_{i,j} \log(p_{i,j}) \tag{2.6}$$

where the sum is taken over the i classes and j output neurons (corresponding to the number of classes), $t_{i,j}$ is the target probability density and $p_{i,j}$ the computed probability density for class i at neuron j. Generally, the MSE is used mostly in regression and reconstruction problems, whereas the categorical cross-entropy is used mostly in classification problems in combination with the softmax function. The softmax function is defined as

$$a_j = \frac{\mathrm{e}^{Z_j}}{\sum_k \mathrm{e}^{Z_k}} \tag{2.7}$$

Here, a_j is the activation of the jth output neuron,Z_j the weighted input of the neuron and the denominator sums up over all k output neurons. The softmax function effectively squashes a k-dimensional vector containing real values into a k-dimensional vector whose values range from 0 to 1 and sum up to 1. Hence, the outputs of the softmax function can be considered as posterior class probabilities in a classification setting.

Regardless of the particular cost function used, a method is needed to relate changes of the cost function to the parameters of the network. This is provided by the backpropagation algorithm (BP) which is at the core of training neural networks as it provides the partial derivative of the cost function with respect to any parameter (weights or biases) $\frac{\partial C}{\partial w}$, which can subsequently be used to update the network parameters. The following discussion is based on [8]. The BP algorithm can be explained using four equations. The first equation expresses the error of each neuron in the output layer as

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{2.8}$$

where a_j^L is the activation of the jth output neuron and $\sigma'(z_j^L)$ is the derivative of the activation function at the weighted input z_j^L. (Note that this error is used only as an

intermediate quantity and is not of importance on its own.) This equation can be put in matrix form as in

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{2.9}$$

where $\odot$ denotes the Hadamard product (element-wise multiplication). The second equation expresses the error of a layer as function of the error of the next layer:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{2.10}$$

where $(w^{l+1})^T$ is the transpose of the weight matrix connecting the $l+1^{textth}$ layer with the l^{th} layer. Combining (2.9) and (2.10) enables computing the error for any layer in the network. The third equation expresses the rate of change of the cost with respect to any bias in the network:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^L \tag{2.11}$$

Thus, the error δ_j^L is exactly equal to the rate of change of the cost function with respect to the bias and it can be readily obtained as shown above. The fourth and final equation expresses the relation between the rate of change of the cost with respect to any weight in the network:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{2.12}$$

The partial derivatives for a given weight are computed using the incoming activations and the output errors which are readily available by now. The backpropagation algorithm thus traverses the network backwards by computing the errors layer by layer and provides the gradient which can then be used to update the parameters to minimize the cost function. The above discussion assumes a single training example has been presented to the network to compute the gradient but the same principle holds for using multiple training examples and using the mean of the computed gradients for the updates.

With the gradient available, the cost function can thus be minimized and the network can start learning. In practice, there are three general ways of learning: online, batch or stochastic learning. The online learning setting means updating the network parameters after each training example and assumes a dynamic environment with a continuous stream of inputs. In batch learning all examples of the fixed training set are propagated forward through the network before computing the gradient of the cost function and updating the parameters. Lastly, there is the stochastic setting, in which the network is updated each time after each example, but again there is a fixed training set. In batch mode, the estimation of the gradient of the cost function is very accurate, whereas in the stochastic mode the estimate is rather noisy. This stochastic behavior can however be very beneficial in avoiding or escaping from local minima, saddle points and plateaus of the cost function. A compromise between computing the true gradient and a very noise estimate can be made by computing the gradient using more than one example, but not using the whole set, a so-called mini-batch. This often results in smoother convergence and can significantly accelerate the training compared with the pure stochastic setting due to a more efficient computation by making use of vectorization. Gradient based

optimization using single examples or mini-batches are both known as stochastic gradient descent (SGD) and combined with backpropagation it is the de facto standard of training in neural networks. The standard SGD method updates the parameters according to

$$\theta_k = \theta_{k-1} + v_k \tag{2.13}$$

where the parameter values at time step k are denoted by θ_k, v_k is the velocity at time step k with which to update the parameters. This velocity is obtained as

$$v = \lambda \nabla_\theta \tag{2.14}$$

where λ is the learning rate and ∇_θ is the gradient of the cost function w.r.t. the parameters θ.

Many improvements on the basic SGD algorithm have been proposed, which include using a learning rate schedule where the learning rate decreases over time with the number of training iterations. The learning rate can also be adjusted based on the gradients as used in RMSprop [9], Adam [10], AdaDelta [11] and AdaGrad [12]. These adaptive gradient algorithms allow a different learning rate for each parameter. It is also common to apply momentum methods which smooth over previous update steps and can strongly accelerate the convergence. The velocity or update step is obtained as

$$v_k = v_{k-1} m - \lambda \nabla_\theta \tag{2.15}$$

where m is the momentum factor which determines the amount of smoothing (0.9 is a typical value) and v_{k-1} is the velocity at the previous time step. Nesterov accelerated gradient or Nesterov momentum is an effective and popular momentum method [13]. Other methods of minimizing the cost function are employed as well, which are generally variants of (quasi-)Newton methods of which L-BFGS is the most popular. However, with growing network and dataset sizes and hence an increase in the number of variables on which the cost function depends, methods based on second order partial derivatives often become infeasible due to the computational complexity involved [2].

Capacity, Generalization & Regularization

Minimizing the cost function can be considered as training the network to approximate a desired function; this function maps the inputs of the network to the correct outputs. The capability of a network to approximate such a function, which might be very complex, is referred to as its capacity. A high capacity means that the network is able to fit a wide variety of functions. The capacity of a network should be chosen such that it is large enough to fit the complexity of the problem, but not much larger. If the capacity is too low, underfitting occurs: the network will fail to approximate the wanted function and this will present itself in failing to obtain a sufficiently low training error. If the capacity is too high, the network will adapt itself to noise or peculiarities of the training data and it will fail to generalize, to succeed on new, unseen examples. This is called overfitting. One of the central challenges in machine learning is to find the right balance between underfitting and overfitting.

To prevent overfitting, regularization methods are applied. Regularization is any modification applied to a learning algorithm that is intended to reduce its generalization error but not its training error. Actually, regularization can hurt training performance. Many regularization strategies exist. There can be hard restrictions on the parameter values, or extra terms added to the cost function that correspond to soft restrictions on the parameter values. The constraints and penalties that are applied can be used to encode prior knowledge or express a generic preference for simpler models in order to promote generalization. In deep learning, the network usually has a very large number of parameters and a high capacity. This also means that deep learning algorithms require a large training set or strong regularization to either reduce the effective capacity of the model or guide the model toward a specific solution using prior information, or both. In practical deep learning scenarios, it is often found that the best model (in terms of generalization error) is a complex model having a large number of parameters that are not entirely free to span their domain [2]. A number of the most common regularization methods will now be discussed briefly.

L^2 Parameter regularization
Penalizing the L^2-norm of the parameters, commonly known as *weight decay* forces the network to have weights with small values. This is usually implemented by adding the sum of the L^2-norms of the weights to the cost function multiplied by a constant which determines the relative importance of the weight decay to the entire cost function. This form of regularization has the effect that the network preferably shrinks weights associated with features that do not contribute significantly to the output.

L^1 Parameter regularization
Penalizing the L^1-norm of the parameters or the sum of the absolute value of the parameters promotes sparseness of the weights, that is, the optimal value of some of the parameters are zero. This can also be considered as a form of feature selection as unimportant features are filtered out by multiplying them with zero-valued weights.

Early stopping
One of the simplest ways to prevent overfitting is to stop training before reaching the minimum training error. The decision to stop training can instead be based on the performance of the network on a validation set. When no longer significant improvements are obtained in an acceptable time, the training is halted.

Data augmentation
The best way for a network to generalize better is to train on more data. Data augmentation is the artificial increase in training data by creating fake data similar to real data. By artificially enlarging the amount of available data the network can capture more relevant information that might lead to better generalization. This strategy can be very useful in applications such as image recognition as new samples can be generated easily by operations such as scaling and rotating the original training data. In other ap-

plications data augmentation might not be possible.

Noise injection
A network that generalizes well will be robust against noise. A natural way of improving a network is then to make it more robust to noise, by adding noise to any of the data, the inputs or activations of the network, the parameters or to the gradient that is used to update the parameters, during training.

Model averaging: bagging & dropout
The combination of information captured by multiple networks can be very effective as well. A bagging approach can be used where different networks are trained using different training sets which are sampled with replacement from the original training set and then combined. However, training many large models from scratch is often impractical or unfeasible.

A very effective alternative yet related method is Dropout introduced by Hinton et al. [14] and treated in more detail by Srivastava et al. [15]. With Dropout a number of randomly selected neurons are disabled during the training of a neural network, so that in effect a smaller subnetwork is trained. See figure 2.2 for an illustration. A

(a) Standard MLP.

(b) After applying Dropout.

Figure 2.2: Dropout applied to a MLP.

subnetwork is trained for a single step and then a next randomly selected subnetwork is trained. Each subnetwork inherits a different subset of parameters from the parent neural network. Through this parameter sharing the subnetworks converge together to a solution and it is possible to represent an enormous number of models using only slightly more resources than needed for the parent model. In the testing phase all these subnetworks are combined as the parent neural network operates normally.

Dropout has proven in many cases to be a simple and computationally cheap yet dramatically effective regularization method. However the effective capacity of the parent network is reduced so it might be necessary to increase the network size, and training the network typically takes two or three times longer [15]. In some cases Dropout seems to offer little benefit or actually hurt performance [16].

2.1.3 Two example neural network architectures

Autoencoder

An autoencoder (AE) is a special form of a MLP with one hidden layer. Instead of predicting some target value (e.g. a class label) it aims to reconstruct its input. To this end it has the same number of output nodes as input nodes. The hidden layer encodes the information from the input and this is decoded at the output layer. When the number of nodes in the hidden layer is smaller than the number of input nodes, the AE is forced to compress or encode the information in order to deliver a good reconstruction of its input. With a larger number of hidden nodes than input nodes, the AE could learn an identity function for each input and effectively just copy the input instead of learning properties about the input. However, the network can be forced to have sparse codes, for instance by penalizing the L^1 norm, so that only a limited number of hidden nodes have a significant activation for a particular input.

Autoencoders can be stacked on top of each other to form deep networks and produce more compressed and abstract representations. It is common to train a deep autoencoder by greedily pretraining the network layer by layer. First, the first layer is trained to reconstruct the input (e.g. an image). Then, the second layer is trained on the activations of the first layer and it is these activations that this layer is trying to reconstruct. The third layer is trained on the activations of the second layer and so on. Any number of layers can be stacked upon one another to construct very deep networks.

Another variant of the original AE was devised by Vincent et al. [17] who propose a denoising autoencoder (DAE). The DAE has the same objective as a regular AE, but instead of directly working on the input data, it works on partially corrupted input data. The DAE corrupts the input data itself with a known corruption process (e.g. additive Gaussian noise, salt-and-pepper noise, masking noise). To obtain a good reconstruction of the uncorrupted input, the DAE has to clean or denoise the corrupted input. To be able to do that, it must learn a feature representation that is robust to noise. This corresponds with the underlying idea that a higher level representation should be robust against noise corruptions of the input. The features that the DAE extracts to denoise its input, are expected to be useful for other tasks as well (e.g. classification). The goal of the DAE is not to be able to denoise input, but to learn better features by using a harder training scheme.

Autoencoders in its many varieties can be used effectively for obtaining compact representations or dimensionality reduction. In [18] feature representations of autoencoders lead to a lower reconstruction error than features representations of equal size obtained by PCA. Autoencoders can also be used to pretrain other networks such as regular MLPs, where the pretraining seems to have both optimization effects, leading to lower training error in the supervised stage, and regularization effects leading to lower generalization error as Erhan et al. show [19]. Paine et al. provide an analysis of the benefits of pretraining [7]. These benefits seem to be absent when large quantities of labeled data are available, however in cases where little labeled data is available pretraining remains a fruitful approach.

Convolutional neural network

The convolutional neural network (CNN) is a neural network model inspired by the mammalian visual system, widely used for image and video recognition. It is designed to take advantage of topological structure of its input. A typical CNN consists of three distinct parts: a convolutional layer, an optional pooling layer, and a final fully connected layer (a typical hidden layer as in a regular MLP).

The convolutional layer
A convolutional layer consists of a set of trainable filters that have a small spatial extent in the width and height dimensions but extends through the full depth of the input volume (3 in case of a RGB image, only 1 for a spectrogram). The input is convolved with these filters, hence the name. Each filter is represented by a neuron that is connected only to a small region of the input image, the local receptive field of the neuron. Applying a filter to the entire image is then equal to having a number of identical neurons which are connected to different parts of the image. The number of neurons needed depends on the size of the filter or equivalently the receptive field size, the stride which is the number of pixels between the centers of neighboring receptive fields in the image and thus determines the amount of overlap between the receptive fields of neighboring neurons, and the use of zero-padding the input around the borders. Generally, 'valid' convolution or 'same' convolution are used. In valid convolution the filter is only applied when it fully overlaps with the image, resulting in a smaller output of $N-k+1$ with N the image size and k the filter size. The disadvantage of this is the relatively rapid shrinking of the image, especially with larger kernels, severely limiting the architectural possibilities. When 'same' convolution is applied, the image is zero-padded such that the output of the convolution is equal to the original image size. Full convolutions, resulting in outputs larger than the original image, are seldomly used in practical CNNs. Optimal padding in terms of classification performance seems to be somewhere in between valid and same convolution [2]. The activation of a single neuron is obtained identical to a neuron in a

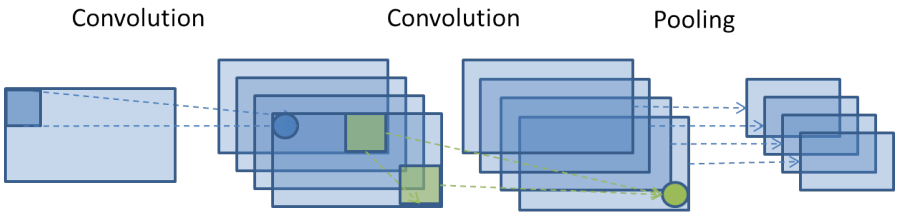


Figure 2.3: Basic convolutional and pooling layer architecture.

MLP (2.1). The output or activation of one filter, which might consist of hundreds or thousands of neurons for the entire image, is called a feature map. A single convolutional layer can have many feature maps. In case when multiple convolutional layers are used in a network, this number of feature maps is considered as the number of channels for the next layer. The receptive field of a neuron in the next convolutional layer is spatially limited in the original image dimensions, but it extends across all the feature maps. This

way, features in higher layers can be very powerful as they combine all the features of the previous layer. See figure 2.3 for a graphical illustration.

The number of trainable parameters of a convolutional layer is equal to the receptive field size times the number of filters plus the biases. This number is often vastly less than the number of neurons which scales proportionally with the size of the input image. These savings in numbers of parameters through the sharing of the parameters is crucial in making large and deep convolutional neural networks feasible.

Pooling layer
CNNs often also contain a pooling layer to condense the information and provide spatial invariance. Each unit in the pooling layer summarizes a $P \times P$ (often 2×2) region in the previous layer by a single output. The most common procedure known as max-pooling lets each neuron from the pooling layer output the maximum activation in its input region. Other procedures exists as well, including average pooling, L^2 norm pooling, weighted average. See chapter 9 of [2] for an overview. Pooling is mostly used to combat overfitting for small datasets. The dimension reduction can also be obtained by using larger strides in the convolutional layer. Since pooling is very aggressive in reducing the size of the representation and is only really helpful in case of smaller data, the trend is to make less use of pooling in modern CNN-architectures.

A CNN can be comprised of many convolutional and pooling layers, but at some point the representation is transformed into vector-form by a regular fully connected layer. Here all the information is integrated and fed to the final part of the network, often a softmax-classifier.

2.2 Radar Fundamentals

Radar (radio detection and ranging) is a measuring principle in which electromagnetic waves with wavelengths ranging from millimeters up to 100 meters are transmitted to generate echoes on distant objects. The transmitted signal can be compared to the received echo signal to establish the object's presence i.e. detection, and to obtain the object's location in terms of e.g. range, azimuth and elevation, as well as the object's radial velocity, by exploiting the Doppler effect. Depending on the radar, a subset of these properties, and sometimes additional properties, can be measured with specific values for sensitivity, resolution and accuracy, all tailored to the application in question. In the radar world, objects of interest are usually referred to as 'targets'. Other objects that generate echoes as well are referred to as clutter. Remaining phenomena that can be found in the received signal are called interference, of which (thermal) noise is always present. The latter provides a ceiling for the radar's performance.

The power received by a radar can be expressed as

$$P_r = \frac{P_t G_T G_R \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{2.16}$$

where P_t is the transmitted power, G_T is the transmitter antenna gain, G_R is the receiver antenna gain, λ is the wavelength of the radar signal, σ is the radar cross section and R is the range of the reflecting object. The radar cross section is the hypothetical area required to intercept the transmitted radar power at the target such that if this intercepted power was re-radiated isotropically, the actually observed power at the receiver is produced. From (2.16) it can be observed that the received power is dependent on R^4 which makes clear that the received power of targets can vary strongly.

The returned signal of a moving target will experience a frequency shift with respect to the transmitted signal which is known as the Doppler effect. The Doppler frequency shift is determined as

$$f_D = -2f_c\frac{v_r}{c} = -\frac{2v_r}{\lambda} \tag{2.17}$$

where v_r is the radial velocity of the object with respect to the radar and defined as positive when the object is moving away from the radar, λ is the wavelength of the radar signal and f_c the corresponding radar carrier frequency. The Doppler frequency shift can be extracted from the received radar signal by a quadrature detector which provides a complex representation of the signal with an in-phase (I) component and a quadrature phase (Q) component. The detector mixes the received radar signal with the reference transmitted signal (the I-component) and with a 90° shift of the transmitted signal (the Q-component) in two separate synchronous detectors. After low-pass filtering, the I and Q outputs are combined to form a complex Doppler signal

$$s_D(t) = I(t) + jQ(t) = \frac{a}{2}\exp[-j2\pi f_D t] \tag{2.18}$$

where a is the amplitude of the received signal. Thus, the Doppler frequency shift f_D can be estimated from the complex Doppler signal $s_D(t)$ by using a frequency measurement tool, such as FFT-based methods or other spectral analysis methods such as AR modeling.

The Micro-Doppler effect

If a target has any part that moves with respect to the bulk motion of the object, then this movement will induce additional frequency modulation of the returned signal. This additional Doppler modulation is called the micro-Doppler effect. The object's size, geometry, configuration and possibly various types of movement (such as rotation, vibration or coning movement) by the body's parts all contribute to the modulation of the signal. Thus, this modulation can be very complex but very distinctive and descriptive as well for any particular object. Hence, the total received modulation is often referred to as the micro-Doppler signature of an object and it can be used for identification and classification purposes.

As the micro-Doppler modulation is caused by parts of the object, the power of the micro-Doppler modulated part of the return signal can be weak and much less than the total returned power which itself can already be weak. In order to obtain a clear view of the micro-Doppler signature, a sufficient signal-to-noise ratio (SNR) is required. It

is possible for a radar to detect an object but the micro-Doppler signature cannot be determined accurately due to a low SNR.

As the modulation due to the micro-Doppler effect can be varying very rapidly in time, to obtain a good characterization of the micro-Doppler signature, it is necessary to have a time-frequency representation. This way, the joint time and frequency information can be better visualized and analyzed. A common time-frequency representation is the spectrogram. This can be provided by the Short-Time Fourier Transform (STFT)

$$S(t,f) = \sum_{\tau=-\infty}^{\infty} s(t)h(t-\tau)\exp(-j2\pi f\tau) \tag{2.19}$$

where $s(t)$ is the time signal to be transformed and $h(t)$ is a window function. The spectrogram is obtained by taking the squared magnitude of the STFT.

$$\text{Spectrogram}(t,f) = |S(t,f)|^2 \tag{2.20}$$

In practice, the STFT is formed by concatenating the regular Fast Fourier transform (FFT) of small, partially overlapping portions of the signal. These signal portions are referred to as time windows or integration intervals. The STFT is faced with a trade-off, as using longer time windows results in higher frequency resolution but reduced time resolution. Vice versa, using shorter time windows provides high temporal resolution but low frequency resolution. Furthermore, the amount of overlap between successive time windows or integration intervals is also an important parameter that must be carefully chosen to obtain a good characterization of the signal with the STFT or spectrogram.

Different kinds of targets can require different processing to obtain an optimal characterization of the micro-Doppler signature. This depends on among others whether the radar echo of a target varies fast with time, frequency or both. The observation or integration interval must be appropriately set for the application, to make sure the variable of interest does not change much during one interval. As an example, the micro-Doppler signature of fast moving objects such as helicopter rotor blades requires short integration intervals. Slower movements allow the use of larger observation intervals for the FFT while providing a clear view of the micro-Doppler signature, as is the case with the human gait movement.

All radar systems that can measure the phase shift of the return signal are in principle suitable for detecting micro-Doppler signatures. These systems include continuous wave (CW), frequency modulated continuous wave (FMCW) and coherent pulse Doppler radars. For FMCW and pulse Doppler radars, the maximum Doppler shift that can be unambiguously resolved depends on a number of radar system parameters. Specifically, the maximum unambiguous Doppler shift can be expressed as

$$f_{D,\max} = \pm\frac{f_s}{2} \tag{2.21}$$

and the maximum unambiguous radial velocity then is

$$v_{u,\max} = \pm\frac{f_s\lambda}{2\cdot 2} = \pm\frac{f_s c}{4f_c} \tag{2.22}$$

where f_s is the Doppler sampling rate, the frequency with which pulses are sent by the radar (or sweeps in case of FMCW), and f_c is the radar carrier frequency. The first factor 2 in the denominator comes from the fact that the Doppler shift can both be positive and negative, and the second factor 2 accounts for the Nyquist sampling criterion. Similarly, assuming small pulse duration the maximum unambiguous range is given by

$$R_{u,\max} = \frac{c}{2f_s} = \frac{\lambda f_c}{2f_s} \tag{2.23}$$

The product of the two variables is fixed and only dependent on the radar frequency.

$$v_{u,\max} \cdot R_{u,\max} = \frac{f_s c}{4f_c} \cdot \frac{c}{2f_s} = \frac{c^2}{8f_c} \tag{2.24}$$

Consequently, all FMCW and pulse Doppler radars have to make a trade-off between the maximum unambiguous range and maximum unambiguous Doppler that they can resolve. CW radars have no range resolving capability but very good Doppler resolving capability, in practice mostly limited by the sampling frequency which generally must obey the Nyquist sampling criterion (compressive sampling is an exception to this but it is out of the scope of this discussion). For the application of human gait classification, the expected maximum target velocities are rather low so Doppler ambiguity is usually not a serious issue.

2.3 Human gait classification

This section discusses the analysis and classification of human gait in radar. First, the human walking motion will be analyzed and the widely used Boulic-Thalmann model for simulating human motion will be described in section 2.3.1. Section 2.3.2 reviews the state of the art of human gait classification in radar.

2.3.1 Description and modeling of human motion

The human walk is a periodic motion with each foot moving from one position of support to the next position of support, periodically swinging the legs and moving the body's center of gravity up and down, and optionally periodically swinging the arms as well. A single cycle of the walking movement can be divided in a stance phase, which accounts for about 60% of the cycle duration, and a swing phase that occupies the rest of the cycle. In the stance phase, one foot is in contact with the ground, while the other foot can be either swinging or touching the ground as well. During the swing phase, the foot is lifted from the ground and the leg is swinging. The left and right stance phases partially overlap resulting in short periods of time where both feet touch the ground at the same time; this is referred to as the double support period. Though every human walk has the same general manner, the individual human gait still carries personalized characteristics, enabling people to recognize a friend from his or her walking style. The human gait may thus be used for personal identification.

Boulic, Thalmann and Thalmann proposed a global human walking model based on empirical mathematical parameterizations derived from biomechanical experimental data [20]. This model is based on averaging parameters from measurements. Hence it is an averaging human walking model without information about personalized characteristics or features of motion. The model provides 3-D spatial positions and orientations of any segment of a walking human body as function of time. Chen [21] provides a radar model based on the Boulic-Thalmann model. In this particular model, the human body is represented by 17 control points which are joint points linking the various body segments. The body segments are modeled by ellipsoids. A spectrogram of the micro-Doppler signature of the modeled human gait is shown in figure 2.4.

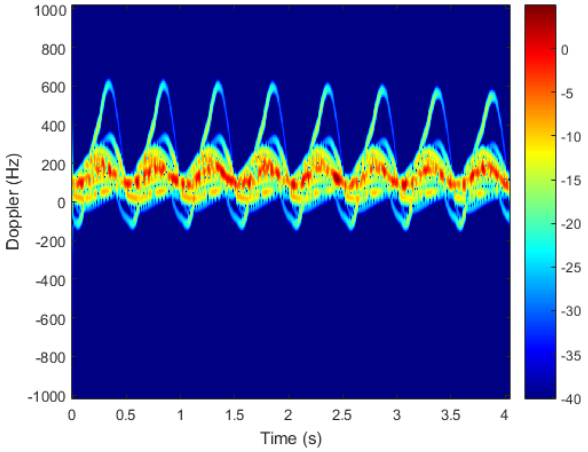


Figure 2.4: Micro-Doppler signature of walking human model. Adapted from [21].

2.3.2 State of the Art Human Gait Classification in Radar

This section reviews the literature concerning the application of human gait analysis and classification based on micro-Doppler signatures in radar. Two main approaches can be discerned in the literature namely parametric and non-parametric. This section concludes with a discussion about challenges and open issues in the research topic of human gait classification.

Parametric classification

Many researchers have used parametric methods for the classification of human gait and human activities in general. In these parametric approaches, explicit predefined parameters are extracted from the micro-Doppler signature and used as features for classification. The early work of Van Dorp & Groen [22] used the previously discussed Boulic-Thalmann model for human motion estimation, and they obtained the model parameters by minimizing a residual error between model predictions and radar mea-

surements. The same authors extracted features from the spectrogram which are then used to estimate the model parameters in [23].

More recently, Guldogan et al. [24] and Groot et al. [25] also used the Boulic-Thalmann model for tracking and classification purposes while making use of particle filtering techniques to estimate the parameters from the spectrogram.

Chen [26] observed that features such as the torso signature, the maximal Doppler shift of the signature, the offset of the signature from the principal Doppler shift, the maximal Doppler variation of the torso line, the oscillation frequency or period of the motion, the kinematic parameters of limbs, among others have been used successfully in classification. Kim & Ling in [27] and [28] used an artificial neural network and a support vector machine respectively to classify human activities based on six similar features. Alemdaroglu et al. [29] analyzed the efficiency of the features used by [27, 28]. They concluded that the torso signature and the offset are the most effective features, whereas the total bandwidth and the bandwidth of the torso signature are moderately effective. The period of a motion was found to be a weakly discriminating feature for similar motions (e.g. crawling and creeping are hard to distinguish from each other, but are distinguished from running), and the standard deviation of the spectrogram above noise levels was found to be nearly useless.

Recently, much effort has been put into extracting individual components of the micro-Doppler signature, for instance by Fairchild & Narayan using techniques as empirical mode decomposition (EMD) to facilitate the parameter extraction and estimation [30]. Orovic et al. [31] used the multiwindow S-method, a special type of time-frequency distribution, in favor of a regular STFT to isolate the contributions of the arms. The swinging motion of the arms is a major contributor to the micro-Doppler signature of the human gait and is recognized as a powerful discriminating feature for activity classification. The works of Tivive et al., Lyonnet et al. and Orovic et al. [31, 32, 33] all involved classifying the type of arm motion to determine more specifically the type of human motion.

Non-parametric classification

The human gait classification problem has also been approached with non-parametric methods which do not consider any explicit predefined model or features, but which can be considered more data driven. Lyonnet et al. [33] computed an average spectrogram for different classes of human activities and performs the classification based on the distance between the test object and the class templates, using a few different distance measures. In the work of Li et al. [34] the techniques of 2D2-PCA and 2D2-LDA were used to obtain subspace representations of portions of the spectrogram centered on the torso contribution.

Tivive et al. [32] used an image recognition inspired approach. A hierarchical method was applied, using fixed directional filters directly applied to the spectrogram, followed by trainable filters and a final classification unit. Very recently, inspired by the successes of Deep Learning networks in the field of image recognition Kim & Moon [35] applied the general convolutional neural network architecture to the spectrogram of micro-Doppler

signatures for classifying human activities. Though failing to outperform their previous results in [27, 28], the results obtained by their convolutional neural network show potential for further applications of Deep Learning based approaches in the radar domain and human gait classification in particular.

Challenges and perspectives in human gait classification

As the research on human gait classification using micro-Doppler signatures only has a short history, challenges and open issues remain [26]. An important challenge is to find the mapping between micro-Doppler signatures and the physical body parts. The first step would be to decompose the micro-Doppler signature into mono-component signatures that associate with single parts, such as the torso, arms, legs and feet. Next, the embedded kinematic/structural information should be extracted from these mono-component signatures. Despite many efforts such as those by Fairchild & Nayaranan [30], Thayaparan et al. [36] and Li et al. [37] these are still considered challenging or even open issues.

Current research topics also include the use of ultrawideband radar and multistatic radar. The use of UWB radar might be beneficial by providing both high range resolution and high Doppler resolution as combining these qualities might lead to better analysis of the micro-Doppler signatures by using a joint range-time-frequency representation. Examples are the work of Wang & Fathy in [38] and Ghaleb & Vignaud in Chapter 4 of [26]. Another interesting research topic is the use of multistatic radar. The work of Tahmoush & Silvious suggested that gender classification is possible using the micro-Doppler signature but performance was strongly dependent on aspect angle [39]. By combining information from multiple radars, the micro-Doppler signatures based on different aspect angles can be used which might improve classification performance, as shown by Fioranelli et al. in [40].

3

Research design

The main objective of this research is to assess the merit of using Deep Learning techniques for the purpose of human gait classification in radar. This chapter presents the research design and methodology of the project to achieve this goal. Section 3.1 presents and motivates the research questions. In section 3.2 the approach that has been used to provide answers to these questions is described. Finally, the data sources and materials used are described in section 3.3.

3.1 Research questions

The review of the state of the art in human gait classification in section 2.3 indicated that good results have been achieved in human activity classification using various methods. However, classification scenarios with multiple targets have received little attention in literature. Multiple targets conceivably pose a major challenge for any classifier and this makes a multi-target classification scenario a challenging and interesting test case to assess the usefulness of a Deep Learning approach.

This study considers three different Deep Learning based network architectures (described in section 2.1): the multi-layer perceptron (MLP), the autoencoder (AE) and the convolutional neural network (CNN). By assessing their classification performance and determining the influences of the main properties of these network models on their performance, improved understanding of Deep Learning methods is obtained and better models may be designed.

The main research objective of assessing the merit of Deep Learning techniques is thus achieved by the design and evaluation of Deep Learning based networks for a multi-target human gait classification scenario, guided by the following research questions:

1. What type of neural network is most suitable for the classification using spectrograms?
2. What is the influence of the numbers of layers and neurons per layer (depth and width of the network) on the network's performance?

3. What is the influence of hyperparameters such as learning rate, optimization algorithms, parameter initialization and regularization methods?

4. What is the operational performance of the classifier in terms of classification accuracy, noise robustness, reliability, interpretability, training requirements and computational complexity?

5. How is the performance of the deep neural network classifier compared with other classification techniques, such as the 3-Nearest Neighbor classifier?

Due to the large scope and number of variables involved, it is infeasible to answer the above questions exhaustively and definitely. The focus of the project is on the machine learning aspect while keeping the requirements of practical application in mind as key factors in evaluating the designed networks.

The following key issues are out of the scope of this thesis but still relevant: the aspect angle of radar observation, and the format of the radar data. Because of feasibility constraints, the movements of the measured objects will be limited to radial trajectories both inbound and outbound resulting in (near)-frontal aspect angles. For the same reason, a fixed data format based on spectrograms is chosen, which will be described in detail in section 3.3.2.

3.2 Approach

The design problem and research questions were addressed using a three phased approach. Each phase built on the previous phase and focused on different research questions.

In Phase I the most suitable type of network architecture was selected by evaluating various MLP, AE and CNN models, simultaneously investigating the effects of model width and depth per architecture type. A relatively simple classification problem of human walking movement versus human running or non-human targets was used to have a simple start before proceeding to more complex models and problems.

Next, in phase II a more advanced and deeper model of the selected architecture (CNN) was designed. The capability of the model to classify synthetic multiple human gaits at different radar frequencies and SNR levels was examined. The influence of training data size and the use of transfer learning on the model's performance were investigated as well. Insight into the network's behavior was obtained using a visualization technique called saliency map.

Finally, in phase III the following additions and changes to the network were considered: data augmentation, noise injection, weight initialization and training algorithms. The final updated model was then applied to experimental multiple human gait data to distinguish the number of gaits. Similarly to the preceding phase, the influence of training data size and the use of transfer learning were examined. In order to further enhance the classification performance, multiple, correlated samples were independently classified and their predictions averaged for the final classification output and low-confidence

predictions were rejected. Saliency maps were used to provide insight into the network's behavior.

Figure 3.1 presents a graphical overview of the proposed approach and shows which research questions are addressed in each phase of the project and what data is used. The materials used including the data are described in the next section.

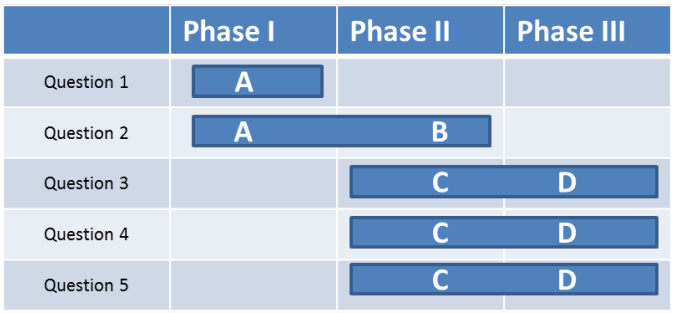


Figure 3.1: Overview of project approach. The letters *A* through *D* in the colored bars refer to the various datasets and different classification problems used, which are described in detail in section 3.3.3.

3.3 Data sources and materials

The measurement setup for the radar experiments is described in section 3.3.1. Section 3.3.2 treats the preprocessing applied to all data, while the four different sets of data that have been acquired are described in detail in 3.3.3. The software and hardware used for implementing, training and testing the neural networks are described in 3.3.4.

3.3.1 Experimental setup

An X-band CW radar was used to perform the measurements. The radar frequency was 10 GHz and the radar beam was horizontally polarized. Figure 3.2 shows the radar hardware. The radar baseband signal was sampled using the audio line-in input jack at a rate of 8 kHz, stereo (for I and Q channels) at 16-bit precision. Measurements have been performed at the parking space behind the Thales Delft office and at various locations in the city of Delft. In figure 3.3 the parking space and its immediate environment are shown. All measurements of human gait have been performed here. The test subjects were instructed to walk back and forth in a straight line between two designated points, approximately 5 and 35 meters away from the radar. The measurements of other objects not containing human gait have been made at a few different locations in the city of Delft, near the university campus.

3.3.2 Preprocessing

The signal was decimated to 2 kHz, this rate is sufficient for target velocities of up to ± 15m/s which is enough for the human gait signature, and the signal was high-pass

Figure 3.2: X-band CW radar.

Figure 3.3: Measurement environment.

filtered to suppress static clutter. The radar simulations produced signals which were sampled with a rate of 2 kHz. Subsequently, by using the STFT with a 128-point FFT, Hamming windowing and an overlap of 90% between the subsequent integration intervals, spectrograms of size 128 x 192 Doppler and time bins were obtained, corresponding to a time duration of 1.25s. An example spectrogram of a single human gait is shown in figure 3.4. This specific format has been chosen to strike a balance between human

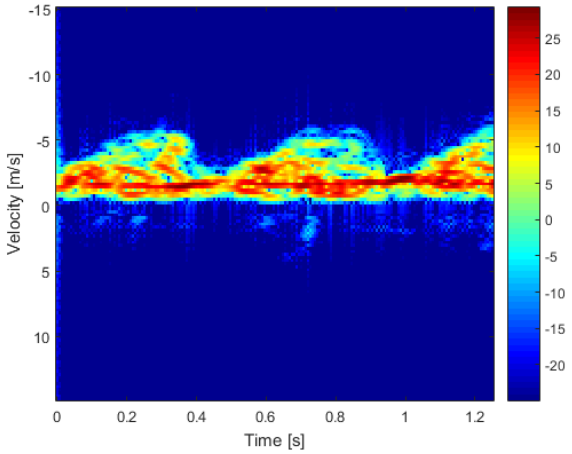


Figure 3.4: Spectrogram of measured single human gait.

interpretability and resolution of the spectrogram on the one hand, and computational complexity and application considerations on the other hand. The time duration of 1.25s was chosen because of two reasons. From an application point of view, the shorter the required observation time of a target, the better. However, to obtain a good characterization of the human gait signature, it is desirable to have at least one full cycle of the periodic human gait movement included in each spectrogram. The cycle duration typically is approximately 1 second for normal walking. In order to have a small margin, we settled on a value of 1.25s. The number of frequency bins used is 128, equal to

the number of FFT-coefficients. The overlap between the integration intervals was then adjusted to compromise, settling at 90%. For the human eye, overlap values of 95%-98% would have been preferable but this would have led to spectrograms at least twice the size and correspondingly increased computational cost.

The spectrograms were in logarithmic scale and these values were then used as linear/gray-scale values. After splitting the data in training and test sets, the spectrogram images were normalized pixel-wise to obtain zero-centered data which is highly advised for data presented to neural networks [41].

3.3.3 Datasets

As mentioned in section 3.2 four different datasets were used whose properties are summarized in table 3.1. Dataset A was acquired by measuring 17 different people engaged in the activities of walking and running. The test subjects varied in age from 20 to 47 years and included 15 males and two females. These measurements were acquired at the parking lot behind the Thales office in Delft at the 25th and 26th of October 2015. Furthermore, targets without human gait were observed in the city of Delft, including objects such as cars, trucks, cyclists, and birds and rustling tree leaves.

Set	Classes	Training samples	Test samples	Remarks
A (experimental)	Inbound walking	2255	751	15 males, 2 females
	Inbound running	581	194	Age 25 - 47
	Outbound walking	2211	738	Spectrogram size:
	Outbound running	547	183	64×192
	Non-gait	2495	832	
B (experimental)	Walking	4466	1489	15 males, 2 females
	Running	1128	377	Age 25 - 47
	Non-gait	2495	832	Size: 128×192
C (synthetic)	Non-gait	1800	450	2000 simulated persons
	1 human gait	1800	450	Height(m): 1.65 - 1.95
	2 human gaits	1800	450	RF(GHz): 2.4, 3, 5, 7 & 10
	3 human gaits	1800	450	SNR(dB): 0, 5, 15, 25
	4 human gaits	1800	450	Size: 128×192
	5 human gaits	1800	450	
D (experimental)	Non-gait	1729	1506	25 males, 4 females
	1 human gait	8510	1379	Age 18 - 47
	2 human gaits	3504	1448	Size: 128×192
	3 human gaits	3837	1443	

Table 3.1: Datasets.

Set A had five classes: inbound walking, inbound running, outbound walking, outbound running and non-human gait/other. This dataset used only the relevant half of the spectrogram dependent on the direction of the target in order to decrease the computational complexity without decreasing frequency resolution. This effectively halved the computational load.

Dataset B consists of the same measurement data as set A, but it used the normal spectrograms and the direction of movement was not used as a variable. Dataset B therefore had three classes: walking, running and non-human gait/other.

Dataset C consists of simulated data, based on the Boulic-Thalmann model as discussed in section 2.3.1. A dataset containing 2000 examples of single walking persons varying in height between 1.65m and 1.95m was built. The simulated subjects walked both inbound and outbound with respect to a simulated CW radar positioned at 1m in height. The start phase of the gait cycle of the subjects was varied according to a uniform distribution to ensure sufficient variance in the data. A class representing absence of human gait was obtained by using complex white Gaussian noise as radar signal. Multiple instances of randomly selected simulated radar returns with single human gait were added together, before the creation of the spectrograms, to provide classes of multiple gaits. Set C contained 6 classes: 0 through 5 simultaneous instances of human gait. Datasets have been produced at radar frequencies of 2.4, 3, 5, 7 and 10 GHz and SNR levels of 0, 5, 15 and 25 dB respectively. The SNR was determined by the average power of the IQ-signal and subsequently adding complex white Gaussian noise.

Dataset D consists of measured data of single and multiple human gaits or non-gait targets. Measurements were made of 12 new different test subjects who engaged in walking either alone or in groups of two or three people. The test subjects varied in age from 18 to 25 years and included 10 males and two females. These measurements were acquired at the parking lot behind the Thales office in Delft at the 29th of February 2016 and the 11th of March 2016. The subjects were instructed to walk back and forth in a straight line between two designated points, approximately 5 and 35 meters away from the radar. The subjects were walking side by side when walking in groups and their relative positions were varied. The radar had near frontal view. Note that parts of the measurement data on which sets A & B were based, were reused for set D.

3.3.4 Software & Hardware

All the neural networks were implemented using the Theano framework [42], an extensive Python library which compiles the Python code to C++ or GPU code. The experiments during phase 1 were performed using a NVIDIA Tesla C2075 GPU. From phase 2 and onwards, the add-on library Lasagne [43] which builds upon Theano was used for implementing the networks. The GPU was replaced by a NVIDIA GTX 980 Ti GPU which more than doubled available computing power. After a few months, the 980 Ti suffered an hardware failure and was replaced by the slightly faster NVIDIA GTX Titan X GPU. All other processing was done using MATLAB. The toolbox PRTools [44] was used for the PCA-based 3-NN classification of the data.

4

Results

The results of this chapter are structured according to the three phases as outlined in section 3.2. Each section of this chapter will introduce its contents in more detail.

4.1 Phase I: Network Architecture Selection

The first phase consists of investigating three types of neural network architectures: the MLP, the AE and CNN, and to determine which architecture is the most suitable for classifying human gait using spectrograms. For the MLP and AE architectures, the influence of both the depth and width of the networks were investigated, whereas for the CNN only depth was considered. The classification problem is to distinguish running from walking and non-human gait by using dataset A. First, the various network configurations and hyperparameter settings used are described and argued in section 4.1.1. The results of the MLP, AE and CNN are shown and discussed in sections 4.1.2 to 4.1.4. The various results are compared and conclusions are drawn in section 4.1.5.

4.1.1 Network design & configuration

For each architecture, the depth of the networks was varied between 1 to 3 hidden layers of equal width. The width or numbers of neurons per layer ranged from 500 to 5000. These numbers were heuristically chosen to be maximally less than half the input size of the data which equals $64 \times 192 = 12288$ units, as dataset A was used for these experiments. Table 4.1 shows the various configurations tested for the MLP and AE architectures. For the CNNs, a convolutional layer with filter size 5×5 followed by a 2×2 maxpooling layer were considered as one hidden layer, and at the end of each CNN a fully connected hidden layer of 500 neurons was used. Table 4.2 shows the various CNN configurations. As is commonly done in literature, the number of filter maps per layer increased with depth and was heuristically determined to keep the computational budget approximately constant across layers. The filter size was heuristically determined to be 5×5 by considering the size of typical features in the spectrogram, such as the curves

due to the contribution of the legs which are only a few pixels thick, and the amount of data available. The 'valid' convolution was used (see section 2.1.3 for details).

Network	Number of layers	Number of neurons	Number of parameters
MLP(AE)-1-500	1	500	6.1M
MLP(AE)-1-1000		1000	12.3M
MLP(AE)-1-2000		2000	24.6M
MLP(AE)-1-5000		5000	61.5M
MLP(AE)-2-500	2	500	6.4M
MLP(AE)-2-1000		1000	13.3M
MLP(AE)-2-2000		2000	28.6M
MLP(AE)-2-5000		5000	86.5M
MLP(AE)-3-500	3	500	6.6M
MLP(AE)-3-1000		1000	14.3M
MLP(AE)-3-2000		2000	32.6M
MLP(AE)-3-5000		5000	111.5M

Table 4.1: Configurations of MLP and AE architectures.

Network	Number of layers	Number of filters	Filter size	Number of parameters
CNN-1	1	[20]	5×5	27.0M
CNN-2	2	[20,40]	5×5	14.0M
CNN-3	3	[20,40,80]	5×5	4.0M

Table 4.2: Configurations of CNN architectures.

The weights of all networks were initialized using Glorot uniform initialization [45], where the weights are sampled from a uniform distribution with zero mean and the minimum and maximum values are scaled depending on the number of incoming and outgoing activations, which is determined per layer. The hyperbolic tangent was used as the activation function. For the classification, the categorical cross entropy cost function was used, while the MSE cost function was used for the pretraining phase of the AE. The standard stochastic gradient descent (SGD) has been used to train the networks with a learning rate of $\lambda = 0.01$ which was the highest learning rate found by trial-and-error to have the networks converge. In case of the AE, the pretraining was also performed using a learning rate of $\lambda = 0.01$.

The pretraining was performed for 10 epochs, while normal training in all cases was performed for 100 epochs. It was observed that the value of the cost function in the pretraining phase was decreased very quickly using only a handful number of epochs and then the cost quickly stabilized and did not significantly decrease anymore, hence only 10 epochs were used. Likewise, early experiments indicated that 100 epochs of training were sufficient to reach the overfitting regime of the networks.

Regularization was applied by penalizing the L^2-norm of the weights using a constant $\alpha = 1 \cdot 10^{-4}$. Though the use of L^1 norm regularization was considered, early

experiments indicated large increases in computational complexity without offering tangible improvements in classification performance. Therefore, L^1 norm regularization was not used in these experiments. Table 4.3 summarizes the training procedure parameters which were used for all experiments of phase I.

Optimization algorithm	SGD
Learning rate	0.01
Batch size	100
Training epochs	100
Pretraining epochs	10
Initialization	Glorot uniform
L2 regularization	$1 \cdot 10^{-4}$
L1 regularization	N/A

Table 4.3: Training procedure parameters of MLP, AE and CNN.

4.1.2 Multi-Layer Perceptron

Figures 4.1a through 4.1c show the training and test error results of the MLPs with 1, 2 and 3 layers respectively. Table 4.4 summarizes the results by showing the best training error, best test error and worst overfit test error scores of each network. The best performing networks per number of layers are shown in bold. First, the training behavior is discussed.

For MLPs with one hidden layer, wider networks converge faster and achieve a lower training error, though the difference is rather small and diminishes with longer training duration. With 2 or 3 layers, wider networks train slightly slower and achieve slightly lower training errors though the differences are becoming negligible. Note that a few of the deeper and wider MLPs shows some fluctuations in training error compared with monotonic decrease for the one layer MLP.

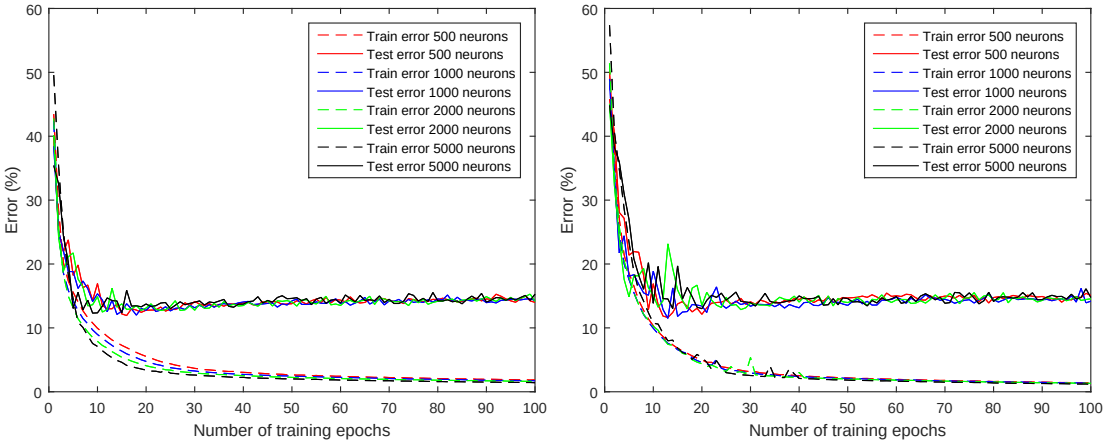
In terms of test performance, the width of a single hidden layer MLP does not matter much: all configurations achieve around 11% best test error and around 14% test error in the overfitting regime. The deepest MLPs also show very similar best test performances for different sizes. However, deeper and larger networks clearly exhibit more variance or less stability in test performance than smaller and shallower networks. The large peaks in the error of the 2000-neuron and 5000-neuron 3-layer MLPs in figure 4.1c in the overfitting regime can be attributed to the fact that the networks suddenly misclassified all the outbound walking examples as inbound walking. Apparently, the class difference is rather small and a small change during the last training epoch led to a hugely different output, demonstrating a highly undesirable lack of stability.

In figure 4.1d the results of the best performing MLP for each number of layers, determined by their lowest test error, is shown in order to compare the influence of the number of layers more easily. The networks achieve very similar test performance, however deeper MLPs take longer to train and the 3-layer MLP clearly is rather unstable.

Apparently for dataset A the depth and width of the MLP do not matter much in the ranges tested except for stability. This makes the smaller and less deep networks the best options, as they perform just as well while having lower computational cost and better stability.

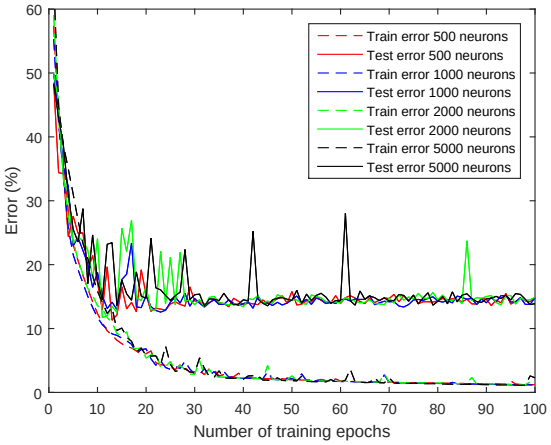
Network	Best train error (%)	Best test error (%)	Worst overfit error (%)
MLP1-500	**1.8**	**11.9**	**15.3**
MLP1-1000	1.7	12.0	15.1
MLP1-2000	1.6	12.5	15.3
MLP1-5000	1.4	12.3	15.4
MLP2-500	1.4	11.6	15.4
MLP2-1000	**1.4**	**11.5**	**16.2**
MLP2-2000	1.3	12.9	15.5
MLP2-5000	1.2	13.0	16.1
MLP3-500	**1.2**	**11.3**	**15.5**
MLP3-1000	1.2	12.6	15.2
MLP3-2000	1.2	11.7	23.7
MLP3-5000	1.1	13.5	28.0

Table 4.4: Performance of MLPs set A.

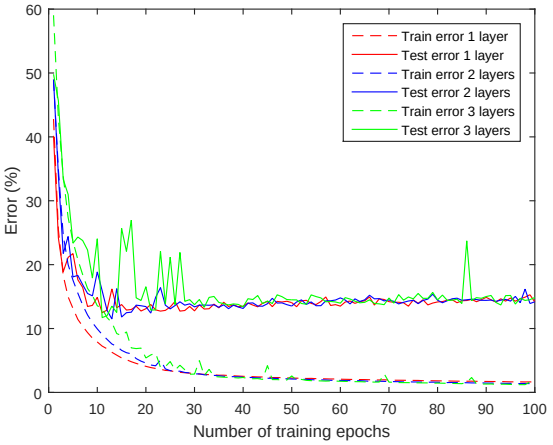


(a) MLP with 1 hidden layer.

(b) MLP with 2 hidden layers.



(c) MLP with 3 hidden layers.



(d) Best MLPs per layer.

Figure 4.1: Performance of MLPs set A.

4.1.3 AutoEncoder

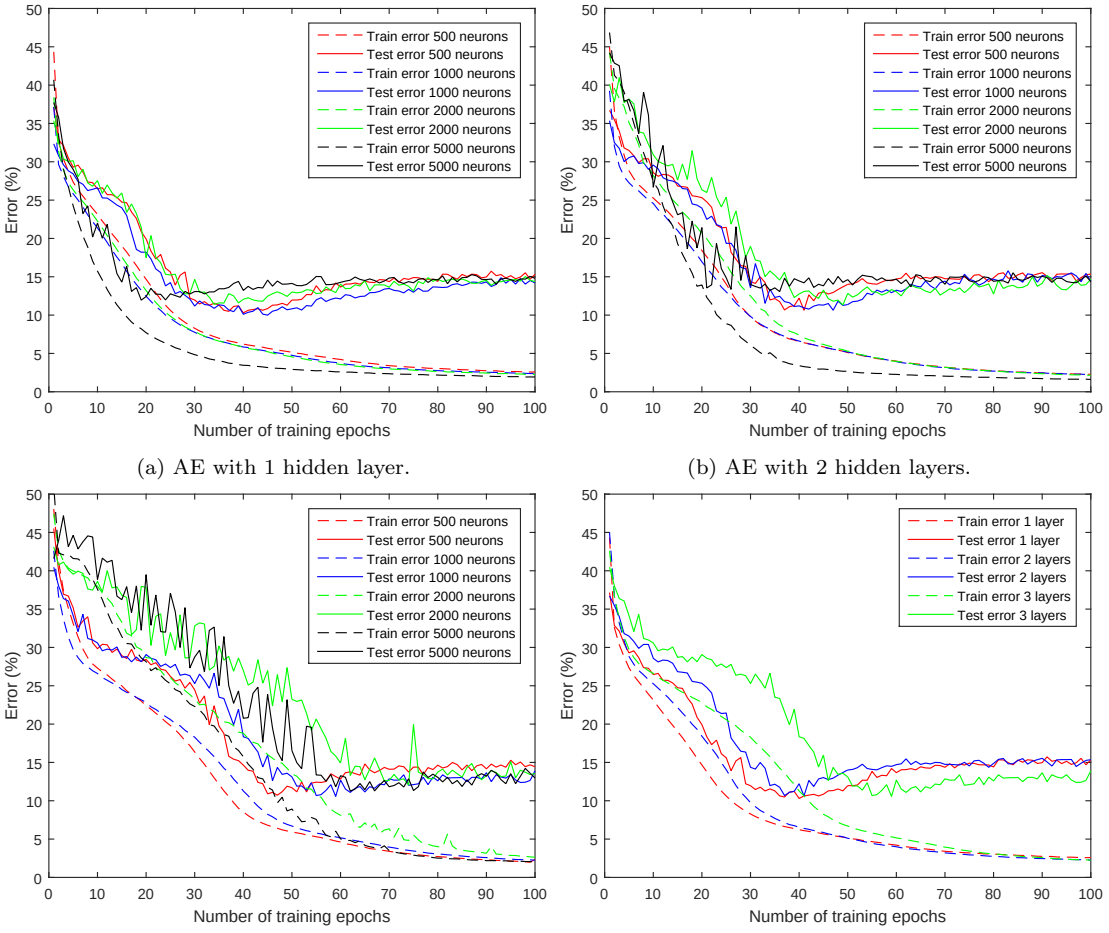
Figures 4.2a through 4.2c show the training and test error results of the AEs with 1, 2 and 3 layers respectively. Table 4.5 summarizes the results by showing the best training error, best test error and worst overfit test error of each network. The best performing networks per number of layers are shown in bold.

In terms of training performance, the AE with one layer and 5000 neurons learned the quickest and best, whereas the smaller single layer AEs all exhibit very similar training behavior and final training error, which was slightly less than the 5000-node AE. For the AE with 2 or 3 layers the relationship between network size and training performance becomes less clear. For the 2-layer network the larger networks obtained lower training error in the end, but the 5000-node network achieved this quickly whereas the 2000-node network only caught up much later. For the 3-layer network smaller networks suddenly achieved lower training error and they did this also more quickly than the wider networks. However the training error decreased much slower in the 3-layer networks compared with the less deep networks although in the end very similar performance was achieved.

The test performance of the AE for various configurations did show a few clear trends. The smaller networks achieved lower best test errors, while the larger networks showed large variability in performance. Test performance in the overfitting regime bears no clear relationship with width of the network and is very similar across all the networks, hovering around 15%. The 3-layer AEs show slightly better performance in this respect, when considering figure 4.2c and ignoring a few peaks which skew the results presented in table 4.5 for the two largest networks. In figure 4.2d the results of the best performing AE for each number of layers, determined by their lowest test error, is shown in order to compare the influence of the number of layers more easily.

Network	Best train error (%)	Best test error (%)	Worst overfit error (%)
AE1-500	2.6	10.3	15.7
AE1-1000	**2.4**	**10.0**	**14.9**
AE1-2000	2.3	11.4	15.0
AE1-5000	1.9	12.0	15.3
AE2-500	2.3	10.6	15.5
AE2-1000	**2.2**	**10.6**	**15.8**
AE2-2000	2.2	11.3	15.2
AE2-5000	1.6	13.1	15.5
AE3-500	2.0	10.7	15.3
AE3-1000	**2.3**	**10.6**	**13.9**
AE3-2000	2.6	12.0	23.2
AE3-5000	2.1	11.3	19.7

Table 4.5: Performance of AEs set A.



(a) AE with 1 hidden layer.

(b) AE with 2 hidden layers.

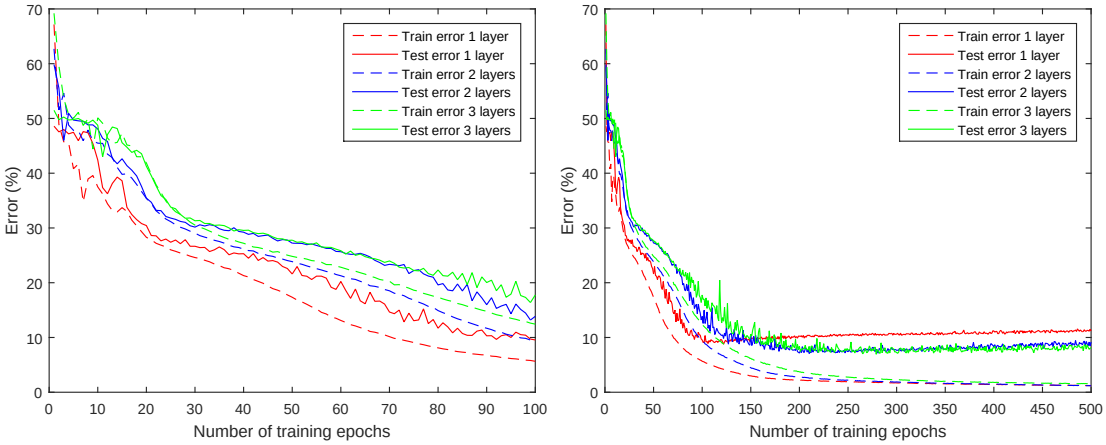
(c) AE with 3 hidden layers.

(d) Best AEs per layer.

Figure 4.2: Performance of AEs set A.

4.1.4 Convolutional Neural Networks

The results of the convolutional neural networks are shown in figure 4.3a. The CNNs needed more training for optimal performance and hence were trained for 500 epochs. The results are shown in figure 4.3b. Table 4.6 summarizes the results by showing the best training error, best test error and worst overfit test error of each network. As suspected, there was a large gain in performance possible by training the networks longer. Deeper networks take longer to train to reach their optimal training error. Two very important observations can be made: the 2-layer and 3-layer networks achieve lower optimal test error, 7.1% and 6.9% respectively versus 8.8% of the single layer CNN, and also clearly suffer less from overfitting than the 1-layer network. The 3-layer network achieves both lowest test error and shows the most resilience against overfitting. Though the deeper networks do show some more variance in test performance than the 1-layer CNN, this variance is quite small.



(a) 100 training epochs.

(b) 500 training epochs.

Figure 4.3: Performance of CNNs set *A*.

Network	Best train error (%)	Best test error (%)	Worst overfit error (%)
CNN-1	1.2	8.8	11.6
CNN-2	1.2	7.1	9.4
CNN-3	1.5	6.9	8.7

Table 4.6: Performance of CNNs set *A*.

4.1.5 Conclusions

To select the best architecture, the best performing network for each architecture were compared in table 4.7. The CNN clearly outperformed the MLP and AE at both best test error and worst overfit error, whereas the MLP achieved the lowest training error. The superior performance of the CNN comes at the price of strongly increased training time and computational cost, however the number of trainable parameters is actually much less than the other architectures. The results clearly indicate that the CNN architecture is a better architecture than the MLP and AE architectures for our classification problem.

Network	Best train error (%)	Best test error (%)	Worst overfit error (%)	Number of parameters
MLP3-500	1.2	11.3	15.5	6.6M
AE1-1000	2.4	10.0	14.9	12.3M
CNN-3	1.5	6.9	8.7	4.0M

Table 4.7: Performance best networks per architecture.

We conclude that the answer to research question (1) (section 3.1, page 21) is the CNN. Our results agree with the consensus in literature, where CNNs have been dominant in many computer vision benchmarks [1] since the landmark publication of [4]. Note that these results were obtained on spectrograms of size 64×192, while preferably larger spectrograms of size 128×192 will be used for further work. This would double the already large number of parameters for the MLP and AE. The CNN architecture is due to its parameter sharing much more scalable to larger inputs. Based on these results and considerations, the CNN is selected as the basic architecture to use for the remainder of this thesis.

The influence of the depth and width of the network depends on the type of architecture. For the MLP, adding more layers and more neurons did not have any benefit for performance, while making the network somewhat unstable and increasing the computational cost. For the AE, the influence seems mixed. Generally, the smaller AEs are slightly superior to the larger ones though this advantage is small and decreases with depth. Remarkably, the 3-layer AEs seem somewhat more resilient to overfitting than the smaller networks. As with the MLP, larger and deeper AEs become less stable but this effect is not as strong as with the MLPs. For the CNN, the effect of depth is clear. Deeper CNNs perform better than shallow ones, though the extra benefit of the third layer is much less than that of the second layer.

To summarize:

- MLP: increasing depth and width leads to less stability without significant effect on test performance.
- AE: smaller networks perform slightly better, deeper networks suffer less from overfitting.
- CNN: deeper networks perform better.

4.2 Phase II: Deep CNNs applied to synthetic data

In this phase, a deep CNN was designed based on literature, the results of section 4.1, and new experiments using the walking versus running versus non-gait classification scenario. In this case, the full-size spectrograms of dataset B were used and the results are described in section 4.2.1. This resulted in a network design denoted as DCNN-1 which was then applied to dataset C, containing synthetic data of multiple human gaits, to investigate whether the new classification problem of distinguishing the number of human gaits could be effectively solved. In section 4.2.2 the DCNN-1 is tested on data based on various radar frequencies and SNR levels, to investigate the versatility and noise robustness of the network. The effect of training dataset size was investigated in section 4.2.3, and transfer learning was investigated in 4.2.4. The network's behavior is visualized in section 4.2.5 and conclusions are presented in section 4.2.6.

4.2.1 Network design & configuration

Various CNN configurations were considered which differ in the number of convolutional layers using filter size 5×5 and 2×2 maxpooling layers. Table 4.8 lists the various configurations. One reason for these configurations was to investigate whether even deeper networks could be beneficial. In section 4.1.4 the three-layer CNN outperformed the shallower CNNs, but the advantage of the third layer over the second layer was much less than that of the second layer over the first. The reason for having two or more convolutional layers in direct succession before applying pooling, as is used in some configurations, is to create more powerful, composite features before the spatial resolution is reduced. Because of the reduction in size of the data traversing the networks due to both pooling and valid convolutions, no more than four pooling layers could be used.
All CNNs had the following properties in common:

- Activation function: Rectified Linear Unit
- Training algorithm: SGD with 0.9 Nesterov momentum
- Two fully-connected layers of 500 and 100 units respectively using Dropout (p=0.5)

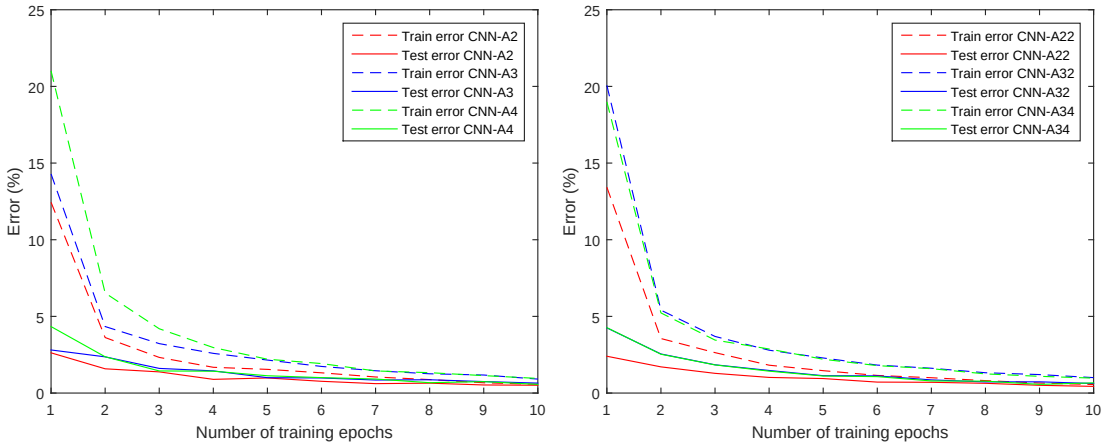
Network	Nr. layers	Nr. of parameters
CNN-A2	[Conv(32) P]*2 FC FC Softmax	21.0M
CNN-A3	[Conv(32) P]*3 FC FC Softmax	4.0M
CNN-A4	[Conv(32) P]*4 FC FC Softmax	0.7M
CNN-A22	[Conv(32)*2 P]*2 FC FC Softmax	17.7M
CNN-A32	[Conv(32)*2 P]*3 FC FC Softmax	2.7M
CNN-A34	[Conv(32)*4 P] [Conv(64)*2 P]*2 FC FC Softmax	4.6M

Table 4.8: Configurations of deep CNN architectures. Numbers in parentheses indicate number of filters for each convolutional layer.

Optimization algorithm	SGD + Nesterov momentum (0.9)
Learning rate	$1 \cdot 10^{-2}$
Batch size	100
Training epochs	25
Initialization	Glorot uniform
L2 regularization	$1 \cdot 10^{-5}$
L1 regularization	N/A

Table 4.9: Training procedure parameters of deep CNNs for set B.

The various CNNs were all tested on dataset B, the training procedure parameters for this experiment are shown in table 4.9. From figure 4.4 it is clear that the different configurations all perform very well and train very quickly. Apparently, the classification problem of set B was not hard enough to distinguish between the various networks. Interestingly, the training error was higher than the test error in this experiment and this was due to the use of Dropout. An important result from this experiment was that even the very deep network employing more than 10 layers could successfully be trained. The deepest CNN has only $4.6M$ trainable parameters despite the deepest convolutional layers having a higher number of filters, while a MLP with a single hidden layer of 500 neurons would have more than $12M$ parameters. Note that the bulk of the parameters of a CNN are those of the first fully connected layer. Since our interest in especially deep networks, the network configuration called CNN-A34 was selected for further use as it has the most trainable nonlinear layers. The selected network is referred to as DCNN-1 in the remainder of this thesis.



(a) CNNs A-2, A-3 and A-4.

(b) CNNs A-22, A-32 and A-34.

Figure 4.4: Performance various CNN configurations on set B.

4.2.2 Multiple human gaits: synthetic case

The DCNN-1 was tested on a harder 6-class problem of dataset C, which contains single and multiple simultaneous instances of simulated human gait, and a non-gait class. Table 4.10 shows the training procedure parameters for this experiment. The learning rate had to be lowered significantly to $1 \cdot 10^{-4}$ and the number of training epochs had to be increased significantly compared with the experiment of section 4.2.1. The results are shown in figure 4.5 and the confusion matrix of the best test result is shown in table 4.11. From the confusion matrix it can be concluded that the DCNN-1 is able to achieve a very high accuracy of 96.6%, with the non-gait class and single gait class being classified perfectly. The classes of four and five simultaneous gaits were much harder, still the DCNN-1 obtained more than 83% accuracy for these classes and the number of predicted gaits was never off by more than one.

Optimization algorithm	SGD + Nesterov momentum (0.9)
Learning rate	$1 \cdot 10^{-4}$
Training epochs	500
Initialization	Glorot uniform
L2 regularization	$1 \cdot 10^{-5}$
L1 regularization	N/A

Table 4.10: Training procedure parameters of DCNN-1 for set C.

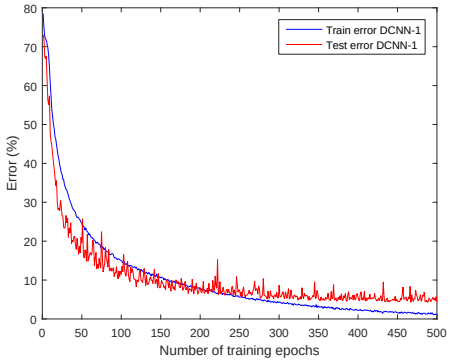


Figure 4.5: Performance of DCNN-1 dataset C (RF = 10 GHz, SNR = 25 dB).

Next, the noise robustness of the DCNN-1 was tested and it was investigated whether the network would work on data based on lower radar frequencies. In dataset C, the difference between lower and higher radar frequencies manifests itself as a vertical scaling of the human gait signature. The signatures differ quantitatively as the Doppler frequencies are diminished proportionally with the relative decrease in carrier frequency (see (2.17)). The results are summarized in table 4.12. As expected, the classification performance diminished with lower SNR levels and lower radar frequencies, but in all cases high to very high accuracies were obtained. The difference between the 25dB and 15dB scenarios (intergroup) is smaller than the variance in best results between models

		Target class					
		C0	C1	C2	C3	C4	C5
Output class	C0	450	0	0	0	0	0
	C1	0	450	2	0	0	0
	C2	0	0	446	5	0	0
	C3	0	0	2	434	10	0
	C4	0	0	0	11	416	32
	C5	0	0	0	0	24	418

Table 4.11: Confusion matrix DCNN-1 dataset C (RF = 10 GHz, SNR = 25 dB).

(intragroup) with the same SNR level. Performances start deteriorating in correspondence with the SNR level not larger than 5dB. The pattern of the errors did not change, almost all of the extra errors in lower SNR scenarios arose in the three, four or five-gaits classes.

To put these results into perspective, a 3-nearest neighbor classifier has been applied to the same datasets. Principal components analysis (PCA) was used to reduce the number of features from 24576 (the number of pixels) to 10 features which accounted for about 85% of the variance in the data. The results of the 3-NN classification are shown in table 4.13. The performance of the 3-NN varied from good to mediocre, and generally was successful at the non-gait, single gait and two-gaits classes while performing quite poorly at the three to five-gaits classes which were often confused with one another. See table 4.14 for the confusion matrix. Generally, the confusion patterns are similar to that of the DCNN-1, but especially for the three-gaits and four-gaits classes the 3-NN has much more errors, and the predicted number of gaits sometimes is two off from the correct number. These results suggest that the classification problem of dataset C is indeed quite hard and the DCNN-1 strongly outperformed the 3-NN in all scenarios.

[%]		SNR (dB)			
		25	15	5	0
RF (GHz)	10	96.1	96.0	91.0	86.1
	7	92.9	91.8	88.8	84.5
	5	91.8	91.7	86.4	82.7
	3	86.9	87.1	86.0	83.5
	2.4	84.9	85.7	81.3	78.2

Table 4.12: Accuracy DCNN-1 dataset C for various RF and SNR.

[%]		SNR (dB)			
		25	15	5	0
RF (GHz)	10	83.8	80.6	74.9	65.2
	7	67.3	67.1	69.8	58.5
	5	56.9	59.7	61.6	59.3
	3	57.7	58.3	54.7	50.5
	2.4	52.1	52.9	53.1	49.2

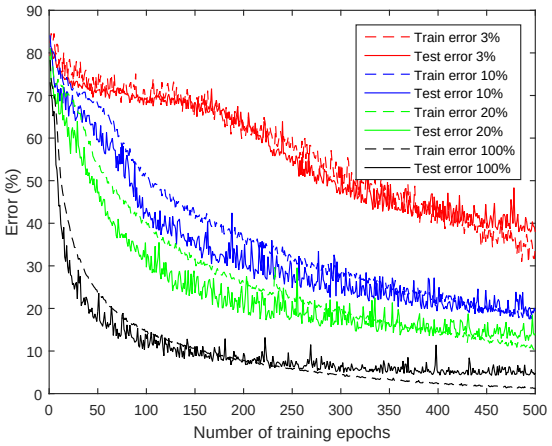
Table 4.13: Accuracy 3-NN dataset C for various RF and SNR.

		Target class					
		C0	C1	C2	C3	C4	C5
Output class	C0	450	0	0	0	0	0
	C1	0	437	8	0	0	0
	C2	0	13	407	57	2	0
	C3	0	0	34	315	105	12
	C4	0	0	0	74	233	104
	C5	0	0	0	4	110	334

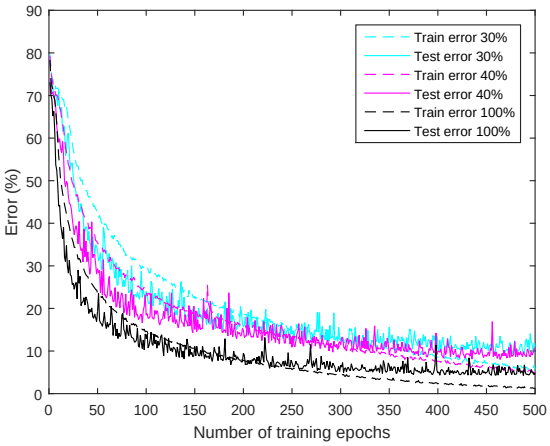
Table 4.14: Confusion matrix 3-NN dataset C (RF = 10 GHz, SNR = 25 dB).

4.2.3 Training set size

In machine learning applications, the amount of available training data is of crucial importance. For the 10 GHz and 25 dB SNR subset of set C it was investigated how the performance of the DCNN depends on the training set size. The results are shown in figure 4.6 for the cases where 3, 10, 20, 30, 40 or 100% of the data is used (100% equals 1800 examples, see table 3.1). The results strongly indicate more training data leads to significantly better train and test performance. Note that the slower training in cases of smaller training sets can be largely attributed to the the correspondingly lower number of parameter updates.



(a) Training set sizes: 3%, 10%, 20% and 100%.



(b) Training set sizes: 30%, 40% and 100%.

Figure 4.6: Effect of training data size DCNN-1 on dataset C (RF = 10 GHz, SNR = 25 dB).

4.2.4 Transfer learning

A possible remedy for the problem of little training data while using complex models such as the DCNN-1, is by making use of transfer learning. In transfer learning, models trained on some task are used as pretrained models for a new task. Here we investigated

whether transfer learning can be used. A few different scenarios were considered, with the source task and target task of the model being relatively similar or dissimilar, and using all or little training data in the second training or fine-tuning stage. Table 4.15 shows the tested configurations and figure 4.7 shows the results.

It can be concluded that the transfer learning was very successful in both scenarios. In case of the 10 GHz source task, the tasks differ only in the SNR level so good results are to be expected. After only a few epochs, the pretrained model reached its top performance, which was significantly better than the model trained from scratch for all training set sizes considered (compare with figure 4.6a). In fact, the pretrained model using 100% of the training data in its fine-tuning stage achieved a best accuracy of 97.4% surpassing the best result achieved without pretraining or transfer learning, shown in figure 4.5. This indicates that with more training data even better results can be achieved.

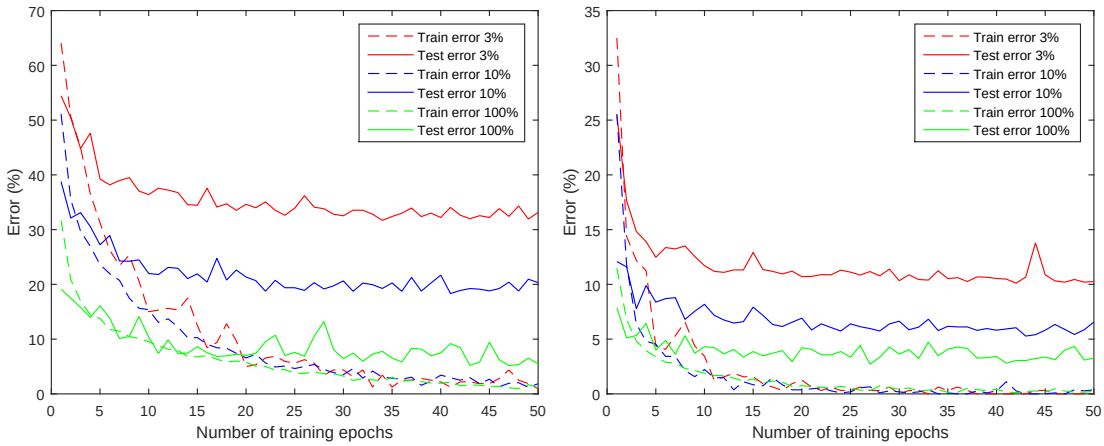
Transfer learning based on the 2.4 GHz source task was also very successful. The training was accelerated enormously and test performance for the smallest training set size was better compared with the model trained from scratch with the same data. The models trained with 10% or 100% of the training set size during fine-tuning performed on par with their counterparts without transfer learning.

Source	Target	Training set size fine-tuning (%)
2.4 GHz 0 dB	10 GHz 25 dB	3
		10
		100
10 GHz 15 dB	10 GHz 25 dB	3
		10
		100

Table 4.15: Transfer learning scenarios DCNN-1.

4.2.5 Visualization of the DCNN

Neural networks are notorious for being black box models where it is hard to gain insight in what they have actually learned. A common method in analyzing CNNs is to plot the filter weights or the activations of the various intermediate layers. A more advanced visualization method has been devised by Springenberg et al. [46] which produces a so-called saliency map. By using the gradients backwards through the network, the pixels of the input image that have a large influence on the classification result are identified. The pixels that by a small variation of their value would lead to large differences in classification get high values, whereas pixels that have little influence on the result when varied get low values. An example spectrogram of four human gaits and its associated saliency map are shown in figures 4.8a and 4.8b respectively. These figures also nicely illustrate the capability of the DCNN-1. At first glance, it seems only three gaits are present in this spectrogram but a close inspection reveals that the outer envelope of the

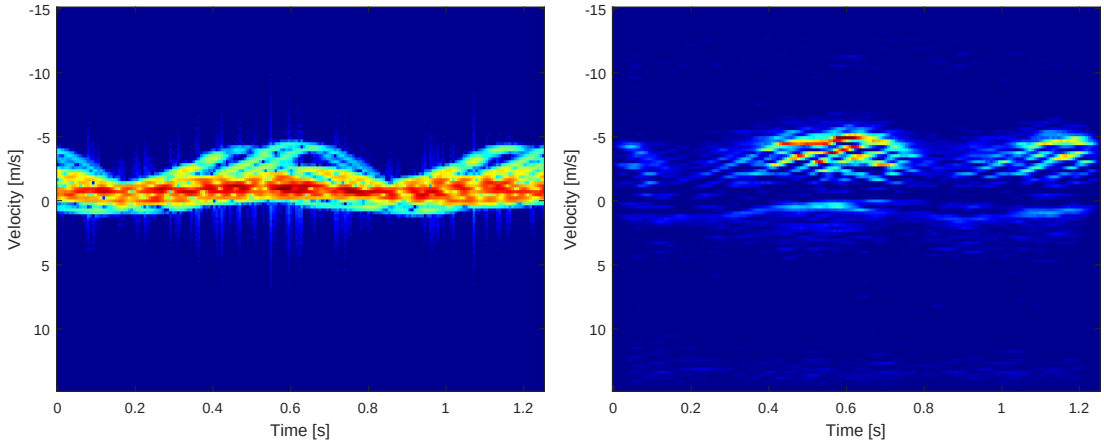


(a) Source: set C 2.4GHz 0dB.

(b) Source: set C 10GHz 15dB.

Figure 4.7: Transfer learning DCNN-1 dataset C (RF = 10 GHz, SNR = 25 dB).

signature in the left-center is slightly wider or broader than the outer envelopes on the right of it. This curve thus represents two gaits which are near-perfectly synchronized in phase. The DCNN-1 classified this spectrogram correctly.



(a) Spectrogram.

(b) Saliency map.

Figure 4.8: Spectrogram and associated saliency map by DCNN-1 of four simulated human gaits (RF = 10 GHz, SNR = 25 dB). Colorbar identical to figure 3.4 for the spectrogram and colorbar ranges [0, 0.1] for the saliency map.

The saliency map shows that a small frequency band is of crucial importance for the classification, especially the region where the contribution of the legs and feet of the human gait signature is located. This is a rather intuitive result. The human gait signature is confined to the low-frequency region of the micro-Doppler spectrum, thus the presence of high frequency components is indicative of a non-human gait instance. To distinguish between the different numbers of instances of human gait, the contribution

of the legs is very important while the contribution of the torso is of limited significance. This makes sense since the torso contribution is extremely similar for all the human gait classes and hence of little discriminatory value. As primarily the contributions of the legs are used for the classification, it becomes clear that the difference in gait cycle phase between persons walking together is crucial for correct classification. If multiple persons move similarly at the exact same time, the leg contributions will also overlap exactly and there will be no clear visual difference in the spectrograms.

4.2.6 Conclusions

The results of this phase clearly demonstrated that it is possible to design and train a deep convolutional neural network for the classification of model based human gait succesfully. We provide preliminary conclusions to the research questions stated in section 3.1.
With regard to research question (2) about the influence of depth and width of the network:

- Many CNN models performed very similar and successfully on set B. Since the models varied strongly in depth and number of parameters, we conclude that the CNN architecture is quite robust and not very sensitive to these differences using this dataset.

With regard to the influence of hyperparameters (research question (3)):

- The learning rate had to be significantly reduced when applying the DCNN-1 to set C. The appropriate learning rate is thus (unfortunately) both model and data-dependent.
- Dropout proved to be an excellent and indispensable regularizer, strongly reducing overfitting and test errors invariably stabilized in the overfitting regime instead of increasing.

With regard to research question (4):

- The DCNN-1 was shown to achieve high accuracy in distinguishing the number of human gaits. The performance decreased as expected with lower SNR levels and lower radar frequencies, but even in the worst case 78% accuracy was achieved. The DCNN-1 was thus quite robust to noise. It must be noted that the classes of three or more gaits were much harder for the model and were the most affected by the misclassifications.
- The amount of training data is indeed crucial for performance, but transfer learning can be used to obtain dramatically increased training speed and on par or better test performance compared with models trained from scratch in case of little training data.

- The visualization of the network's behavior demonstrated that the network learned intuitive features and made clear that the contribution of the legs of the gait signature was crucial in classifying between different numbers of simultaneous human gaits.
- The network needed a lot of training time, generally about 12 hours for most experiments on a single high-end GPU. The computational complexity of training is thus quite high. However, testing the network with a single example takes less than a millisecond on the hardware used, which means that real-time application of the DCNN-1 should be possible for many practical radar systems except for the very low-cost / low-power systems.

With regard to research question (5):

- The DCNN-1 proved to be clearly superior to the 3-NN classifier in all considered scenarios.

The DCNN-1 was thus very successful in distinguishing the number of human gaits. However, these results were obtained on model data and though variability in the spectrograms is obtained by varying the height of the modeled persons, the spectrograms are always very similar. In real life scenarios, the spectrograms of human gait are expected to be much more varied due to clutter, noise, multipath effects, and especially personal variations in gait, effects of clothing, and possibly obstructed view of body parts. In the next section, whether the DCNN can distinguish the number of human gaits using measurement data is investigated.

4.3 Phase III: DCNN applied to measured multiple human gaits

In this phase, the ability of the DCNN to distinguish the number of human gaits was validated using the experimental data of set D. Some small improvements on the DCNN-1 architecture were considered as well: a data augmentation method, a regularization method by injecting noise at the input of the network, a better weight initialization strategy, and more advanced training algorithms. Their effects were investigated in section 4.3.1 and this resulted in a final updated network design denoted DCNN-2 which was used in all subsequent experiments. In section 4.3.2 the effect of training data size was investigated and in section 4.3.3 we investigated the effect of pretraining DCNN-2 using dataset C (synthetic data) before applying it to the measured dataset D. The effect of combining predictions and a rejection option in the classifier in order to enhance the performance were investigated in section 4.3.4. The network's behavior is visualized in section 4.3.5 and conclusions are given in section 4.3.6.

4.3.1 Network design & configuration

First, the DCNN-1 was tested on dataset D. The results are shown in figure 4.9 and table 4.16. The DCNN-1 obtained an accuracy of 85.9% and could easily distinguish the non-gait from the gait classes. The single gait class was classified very well, but the two-gaits class appeared to be rather difficult for the network, overlapping significantly with both the single gait and the triple-gait class.

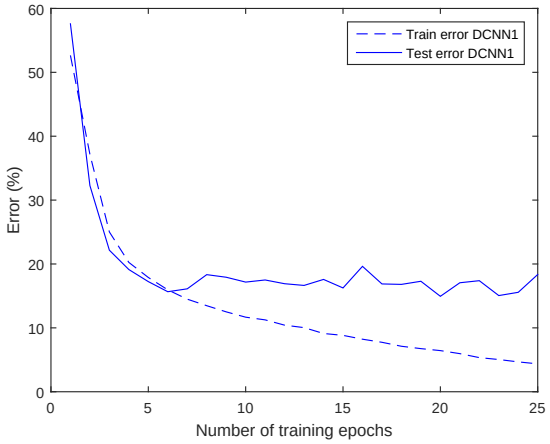


Figure 4.9: Performance DCNN-1 set D.

A common method to improve neural network models is to artificially enlarge the training dataset which is called data augmentation. Common data augmentation methods include rotations, scaling, translations and elastic deformations. For natural images, the meaning is invariant to these operations, e.g. a cat is still a cat when present at different positions in the image or somewhat stretched etc. For spectrograms, the posi-

		Target class			
		Non-gait	1 gait	2 gaits	3 gaits
Output class	Non-gait	1495	0	0	1
	1 gait	3	1348	209	13
	2 gaits	1	29	906	222
	3 gaits	7	0	328	1185

Table 4.16: Confusion matrix DCNN-1 dataset D.

tion of features actually encode properties of the target, thus these methods cannot be applied. However, horizontal translations of the entire spectrogram would correspond to a shift in time, but not qualitatively change the meaning of the spectrogram. Therefore, a custom layer was implemented in Lasagne which shifted and wrapped around the spectrograms along the time dimension a random number of pixels, an operation equal to the 'circshift' operation in MATLAB, during training. This way, the network should learn even more time-translation invariance. The results are shown in figure 4.10. The effect of the horizontal translation is generally a small drop in test performance, while not appreciably impacting the training dynamics. The spectrograms are generally not perfectly periodic and the translations resulted in many cases in a sort of phase jumps in the gait cycle, see figure 4.11 for an illustration. The resulting images cannot naturally occur in the test set, so the network might have wasted its capacity on learning features that are by definition not realistic.

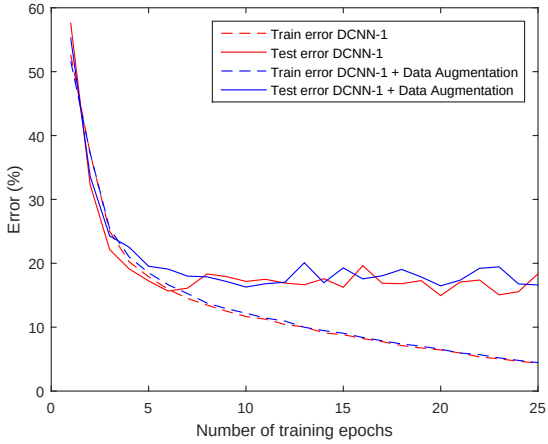
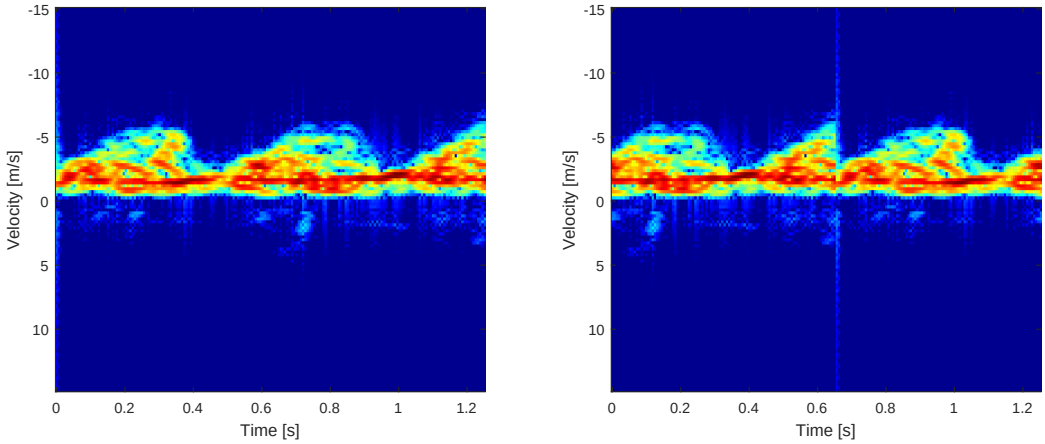


Figure 4.10: Performance DCNN-1 set D with 'circshift' data augmentation layer.

Injecting noise at the inputs of the network is a common regularization strategy. A layer adding zero-mean Gaussian noise to the input data was added to the network. The results are shown in figure 4.12. The noise injection resulted in very small improvements in test performance. The magnitude of the injected noise does not make much difference,

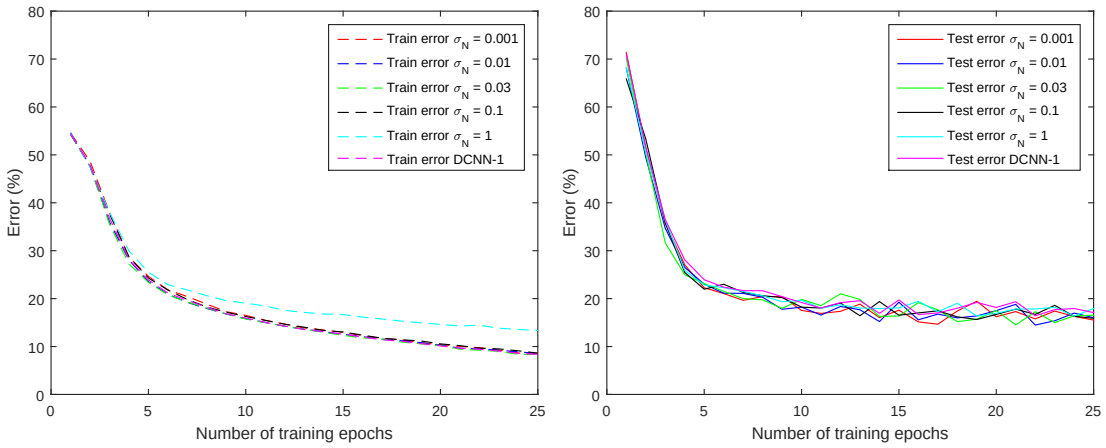


(a) Spectrogram of single human gait.

(b) Horizontally translated spectrogram of single gait exhibiting gait phase jump.

Figure 4.11: Spectrograms of single human gait with and without horizontal translation. Color bars identical to figure 3.4.

except for the highest noise level where the noise injection lead to much slower training, but comparable test performance.



(a) Training scores.

(b) Test scores.

Figure 4.12: Performance DCNN-1 with noise injection.

The initialization of the network weights is of crucial importance, as suboptimal initialization can lead to divergence of the network or significantly lower performance. Glorot uniform distribution initialization (which has been used in all experiments hitherto) was compared with the He Gaussian distribution initialization [47]. The latter initialization has incorporated an extra scaling which is specifically derived for the ReLU activation function, while the Glorot initialization was based on an analysis assuming identity functions. The results are shown in figure 4.13. The He Gaussian initialization

achieved a slightly better test error than the Glorot uniform initialization, while training a bit slower.

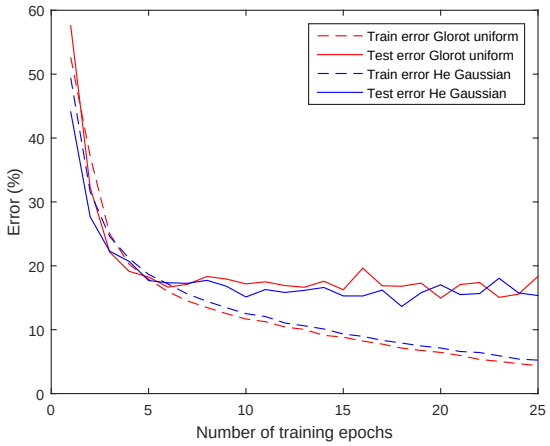


Figure 4.13: Performance DCNN-1 with Glorot uniform or He Gaussian weight initialization set D.

The following optimizers (training algorithms) were compared: SGD with Nesterov momentum [13], RMSprop [9], Adam [10] and Adadelta [11]. A learning rate of $\lambda = 1 \cdot 10^{-4}$ was used in all cases and the results are shown in figure 4.14. Apparently, the learning rate was a bit too high for Adadelta to deliver satisfactorily performance. The best test performance was rather similar for the other considered optimizers, specifically with Adam and RMSprop both slightly outperforming SGD with Nesterov momentum. RMSprop showed a bit more variance in test performance than Adam, and it also trained slightly slower. The difference between Adam and RMSprop is however rather small while these adaptive learning methods both clearly learn much faster than SGD with Nesterov momentum.

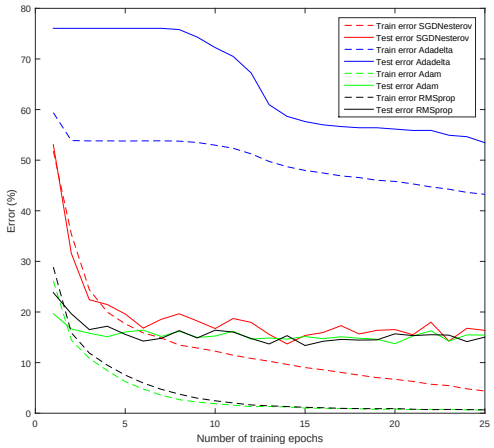


Figure 4.14: Performance DCNN-1 using SGD-Nesterov, RMSprop, Adam and Adadelta training algorithms set D.

Considering the above results, the final network design, denoted as DCNN-2, incorporated the Gaussian noise injection layer, the He initialization strategy and Adam was chosen as the training algorithm. This configuration was used for all remaining experiments. Table 4.17 summarizes the training procedure parameters for the DCNN-2. The results of the DCNN-2 are shown in figure 4.15 and table 4.18. The DCNN-2 achieved 87.0% accuracy and thus outperformed the DCNN-1 though the difference was rather small. The noise injection increases the computational cost slightly, while the adaptive training algorithm increases the computational cost per epoch significantly. However, the accelerated training because of the adaptive algorithm leads to fewer training epochs and this outweighs the extra computational cost by a large margin. The DCNN-2 can thus be considered as a small but satisfying improvement over the DCNN-1. The number of human gaits classification problem was also classified using the 3-NN classifier. The results are shown in table 4.19. The 3-NN classifier performed very well on the non-gait and single gait classes, but performed poorly on the two-gaits class and mediocre on the three-gaits class, supporting the previous observations about the overlap between the classes. For the 3-NN, the overlap of the two-gaits class with the single gait class was much larger than for the DCNNs, and also the overlap of the three-gaits class with the single gait class. This might be caused by the class imbalance in the training set where the single gait class is much larger than the other classes (see table 3.1).

Optimization algorithm	Adam
Learning rate	$1 \cdot 10^{-4}$
Batch size	10
Training epochs	25 - 50
Initialization	He Gaussian
L2 regularization	$1 \cdot 10^{-5}$
L1 regularization	N/A

Table 4.17: Training procedure parameters of DCNN-2.

		Target class			
		Non-gait	1 gait	2 gaits	3 gaits
Output class	Non-gait	1501	0	0	0
	1 gait	3	1337	139	9
	2 gaits	2	39	939	185
	3 gaits	0	1	368	1227

Table 4.18: Confusion matrix DCNN-2 set D.

		Target class			
		Non-gait	1 gait	2 gaits	3 gaits
Output class	Non-gait	1501	1	0	0
	1 gait	4	1279	560	145
	2 gaits	2	78	586	502
	3 gaits	1	21	302	786

Table 4.19: Confusion matrix 3-NN set D.

4.3.2 Training data size

The results of DCNN-2 when only using parts of the training data are shown in figure 4.16. For this particular experiment, the large class imbalance in the training set (the single gait class was two to three times larger than the other classes, see table 3.1 for

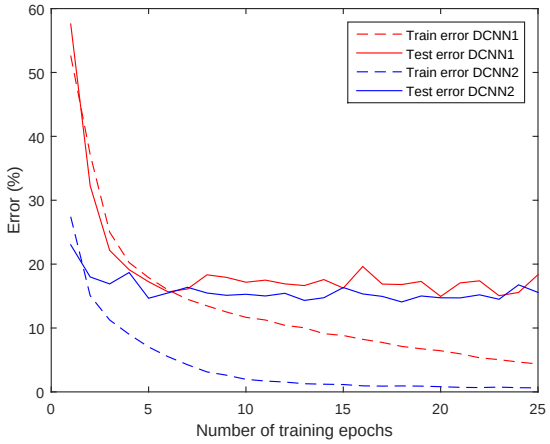


Figure 4.15: Performance of DCNN-1 and DCNN-2 set D.

details) was removed by only using 4000 randomly selected examples of the single human gait class, and from this lower subtotal varying training set sizes were used. As expected,

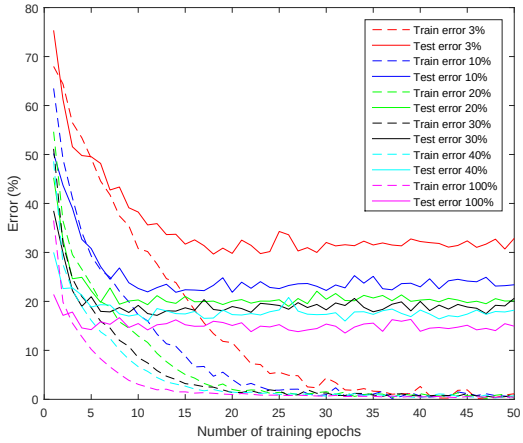


Figure 4.16: Effect of training dataset size DCNN-2 set D.

more data leads to better test performance. Note that the performance of the DCNN-2 using the entire but balanced training set was not significantly affected compared with the entire, imbalanced training set (compare the 100% result in figure 4.16 and figure 4.15). The slower training using the smaller training sets is due to the lower number of parameter updates per epoch.

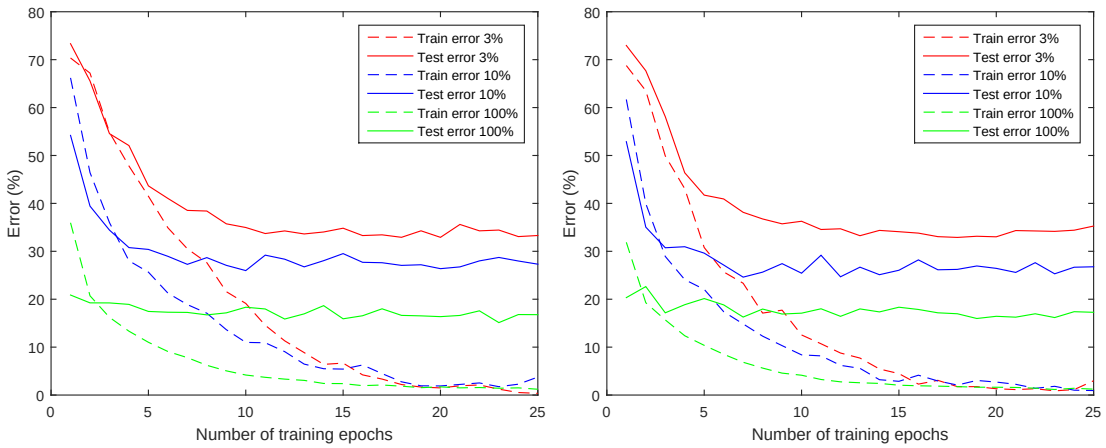
The non-gait class was always classified with high accuracy. The accuracy on the single gait class decreased significantly for training set sizes 10% and 3%, with a worst case of 65% accuracy, but was barely affected in the other cases. The accuracy on the two-gaits class suffered strongly, sometimes dropping to only 40%, whereas the three-gaits class was classified with more than 75% accuracy for the 3% training set.

4.3.3 Transfer learning

Here we investigated whether training the DCNN-2 on dataset C, the synthetic multiple human gait data, is helpful for classifying the measurement data of dataset D. Table 4.20 shows the tested configurations and figure 4.17 shows the results. It can be concluded that pretraining with dataset C resulted in a significantly accelerated training for the smaller dataset sizes but somewhat less test performance for all configurations, including for the full training set size. The drop in test performance was generally about 2% - 3% (compare with figure 4.16). The model pretrained on the 0dB SNR source trained slightly faster than the model pretrained with 25dB SNR data, but the test performance was identical.

Source	Target	Training set size finetuning
		3
Set C: 10 GHz 25 dB	Set D	10
		100
		3
Set C: 10 GHz 0 dB	Set D	10
		100

Table 4.20: Transfer learning scenarios DCNN-2.



(a) Source: set C 10GHz 25dB.

(b) Source: set C 10GHz 0dB.

Figure 4.17: Transfer learning DCNN-2 for set D by pretraining on set C.

4.3.4 Averaging multiple predictions & rejection option

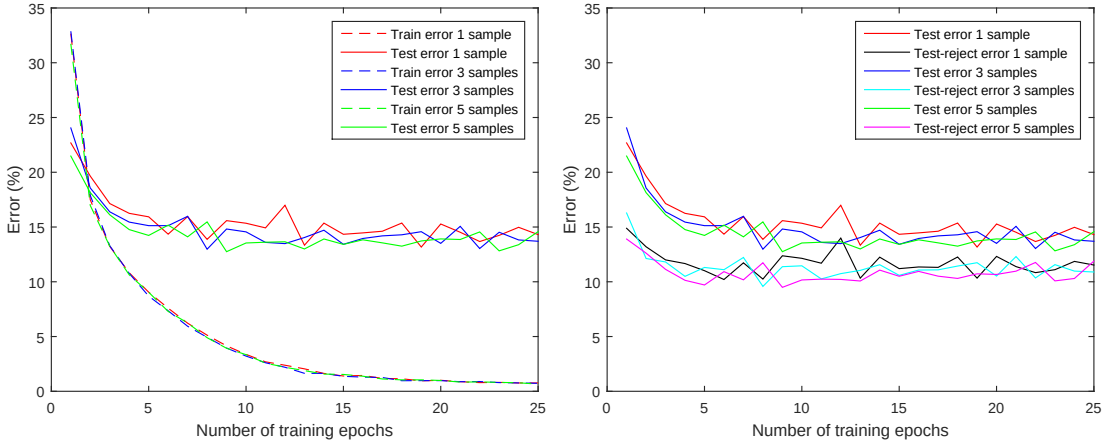
The results from section 4.3.1 indicated that distinguishing between two and three instances of human gait is rather hard. As noted during the experiments with synthetic data, this might be due to synchronization of gait phase between the persons. A straight-

forward possible solution is to combine multiple classification results of consecutive measurements in the hope that the synchronization varies somewhat in time. For the combination of multiple measurements, a number of consecutive measurements were classified independently and the predictions were then averaged. Figure 4.18a shows the results.

Another option is to implement a rejection option in the classifier which rejects samples it cannot classify with at least a certain degree of confidence. Both options were implemented. A sample was rejected when the highest class probability was less than two times as high as the next highest class probability. Results are shown in figure 4.18b.

Averaging results of multiple samples slightly reduced the variance of the DCNN-2 test performance, but barely improved on the best result obtained. Close inspection of the misclassifications revealed that the errors were clustered, that is, many of the misclassified samples were consecutive samples that were usually misclassified to the exact same class. This explains the lack of improvement by averaging the predictions.

The rejection of low confidence samples resulted generally in a 5% lower test error. About 10 - 15% of the samples were rejected and the classification accuracy on rejected samples was about 50%. Of the samples that were not rejected still about 10% was misclassified, showing that the DCNN-2 could make wrong predictions with high confidence.



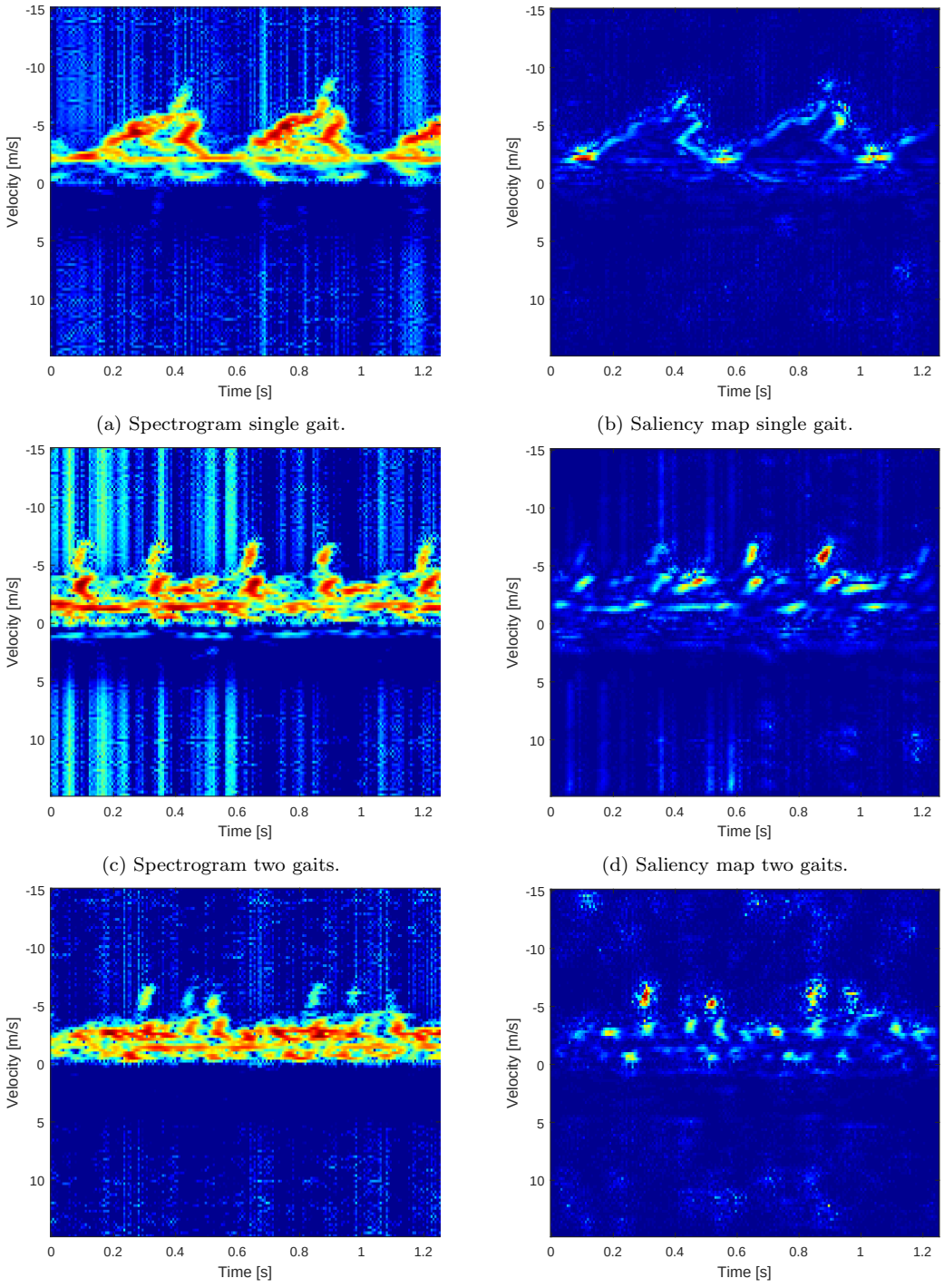
(a) Performance DCNN-2 averaging multiple samples. (b) Performance DCNN-2 averaging multiple samples & rejecting low-confidence samples.

Figure 4.18: Performance DCNN-2 using multiple predictions and rejection set D.

4.3.5 Visualization of the DCNN

The network behavior is visualized using the saliency map. Figure 4.19 shows an example spectrogram for each human gait class and its associated saliency map. The spectrograms show how complex and varied the human gait signature can be compared with the model data (cf. figure 4.8a). Note that the vertical stripes in the spectrograms are caused

by noise and spectral leakage. The appearance of the noise in these spectrograms is strongly exaggerated due to the normalization of the spectrograms. Despite the noise in the spectrograms being of similar magnitude as the gait signatures, the DCNN-2 learned to largely ignore this noise as is apparent from the saliency maps. For the single gait, the network traces the outer envelope of the gait signature. As was noted in section 4.2.5 with the model data, the contribution of the legs is crucial for the classification. The plume-like features which are especially noticeable for the two- and three-gaits classes are intuitively good indicators of the true class and the network has learned this. Note that the contribution of the torso to the signature, which is widely used as feature in human gait classification, is not very important for the DCNN-2. This makes sense, since the torso contribution is very similar for all human gait classes and hence is not distinctive.



(a) Spectrogram single gait.

(b) Saliency map single gait.

(c) Spectrogram two gaits.

(d) Saliency map two gaits.

(e) Spectrogram three gaits.

(f) Saliency map three gaits.

Figure 4.19: Spectrograms and saliency maps of human gait classes by DCNN-2. Colorbar value ranges [-0.5, 2.5] for these normalized spectrograms and [0, 0.1] for the saliency maps.

4.3.6 Conclusions

In this final phase of the project, the experimental multiple human gait classification problem was shown to be satisfactorily solved by a deep convolutional neural network, the DCNN-1 which attained an accuracy of 85%. The upgraded model DCNN-2, by using a more advanced training algorithm, better weight initialization, regularization by noise injection at the inputs and especially by combining multiple predictions and rejecting low-confidence samples, could improve this accuracy to 90%. Generally, the DCNN-2 could distinguish very well between the absence of human gait, the presence of one human gait and the presence of two or more human gaits whereas the distinction between two and three human gaits proved to be quite hard.

With regard to research question (3) about hyperparameters we conclude:

- Data augmentation by random horizontal translations of the spectrograms proved to be counterproductive, slightly hurting generalization error.
- He Gaussian parameter initialization lead to slightly better generalization error but also to slightly slower training than the Glorot uniform initialization.
- Regularizing the DCNN-2 by noise injection during training had no or very small positive effect on generalization error.
- The advanced optimization algorithms Adam and RMSprop both dramatically increased the training speed while having no or a very small positive effect on generalization error with respect to SGD with Nesterov momentum. The appropriate learning rate depends on the training algorithm, as was made clear by the poor results of Adadelta.

With regard to research question (4):

- Large amounts of training data are crucial for performance.
- Using transfer learning by training the DCNN-2 on synthetic data to alleviate the need for training data delivered disappointing results. Though pretrained models trained much faster, their performance was consistently less than models trained from scratch.
- The visualization of the DCNN-2 by saliency maps showed that the relevant features of the signature were intuitively sound.

Finally, with regard to research question (5):

- The DCNN-2 outperformed the 3-NN by a large margin due to its superior performance on the two-gaits and three-gaits classes. These were the most difficult classes for both classifiers.

5

Discussion

In chapter 4, the results were presented and some preliminary conclusions were drawn. A number of results require a more thorough discussion. This discussion is organized according to the three phases of the research project as outlined in section 3.2. Results that do not apply to a specific phase of the project are treated separately. The chapter concludes by discussing some limitations of this work.

Phase I

- *The pretraining of the AE results in much slower training and higher training error but better generalization error compared with the MLP.*

  Since the MLP and AE are identical during the supervised training stage, the difference between these two models is solely due to the different initialization. It is not clear why the training of the AE is slower, though the effect is even more pronounced for deeper networks. An inappropriate learning rate seems implausible, considering the good final performance. For the deeper networks, it might be that the layers learn at different speeds adding to the difficulty of training. These results do agree with the interpretation of the pretraining as a regularization effect instead of an optimization effect, similar to as what was found in [19].

- *The MLP (and AE) best test performance is worse than that of the CNN while showing no convergence problems.*

  This result is actually somewhat unexpected. The large model capacity of the MLP makes it prone to overfitting but theoretically also enables the best performance. Though possibly depending on the radar target, intuitively pixels of the spectrogram are in general locally much more correlated than globally (as is the case with natural images). It could be that the fully-connected neurons have more difficulty in identifying useful features based on the local correlations when at the same time trying to find global structures. In [48] CNNs outperform MLPs of same depth and number of parameters strongly as well on CIFAR-10, a dataset of small natural images. [48] concludes that the convolutional architecture itself is indeed

crucial for top performance. A clear explanation for this phenomenon is however lacking in literature.

- *The mixed influence of the size of the network (depth and width) and model capacity.*

  The rationale of using multiple layers is to obtain more powerful, abstract features. The test performance of the MLP and AE did not vary much with depth, while the training error was decreased slightly for deeper MLPs, but was still higher than zero. Thus a small part of the training data is extremely hard to learn and even huge increases in model capacity (see table 4.1) cannot be leveraged successfully, but the overfitting of all the models was very similar. More abstract features are thus either not helpful on the particular task or the networks failed to learn them.

  It is possible that the lack of improvement of adding more hidden neurons is due to interactions between the neurons. First-order optimization methods (such as SGD and its many varieties, which have been used exclusively in this study) are known to fail in scenarios with many interactions between the neurons [49]. Another possibility is the presence of many more local minima or saddle points in the cost function for networks with more layers and larger capacity.

  For the CNNs, increasing the depth had a clear positive effect. The number of trainable parameters was decreased strongly, though the model capacity is actually increased since it grows exponentially with network depth. Capacity is thus determined both by the number of parameters and the architecture, and in this case increased capacity is beneficial for performance. Overall, these results indicate that for good performance it is vital to use the available capacity effectively rather than having much capacity. This is to be achieved trough both network architecture design and regularization.

Phase II

- *The effect of training set size*

  As expected, more training data leads to better generalization error. Remarkably, in case of smaller training set sizes, the training proceeded slowly whereas quick decrease of training error was expected. This can be partially attributed to Dropout.

  Training while using about 50 examples per class (the 3% set size) proved to be insufficient to learn the three-gaits and four-gaits classes satisfactorily, whereas for the other classes both training and generalization error was reasonable. It is not surprising that the network has the most difficulty learning the intermediate classes, which likely have the most overlap with other classes. The required amount of training data can vary strongly for each class, but is in any case rather large.

Phase III

- *The effects of optimization algorithms.*

  Not surprisingly, the adaptive training algorithms Adam and RMSprop make the DCNN train much faster than using SGD with Nesterov momentum. However the generalization error is barely affected. These results imply the rather limited influence of the learning rate on generalization error, since the effective learning rate for each parameter can vary dramatically for the adaptive algorithms. This contrasts with the general consensus in literature which regards the learning rate as the single most important hyperparameter [2, 41]. Much effort has been spent at devising automatic tuning of the learning rate such as [50] and the adaptive algorithms considered in this work [9, 10, 11]. The observed results are thus likely specific to the dataset, and further research is needed to determine what properties of data have an effect on the learning of a network.

- *The mixed effects of transfer learning.*

  The initial test error (that is after 1 epoch of training) of the model using 3% of the training set is over 70%, as indicated in figure 4.17. Initially, all examples are mapped to the two-gait class. It is unclear why this occurs, but it was observed in all experiments and thus reflects a systematic effect.

  Unfortunately, this result also implies that training the DCNN only on synthetic data and then directly applying it to experimental data will lead to very poor results. No literature has been found which describes a similar setting to compare with. Transfer learning in general has proven to be quite successful and it has become common to use pretrained networks such as Alexnet [4] and apply it to related computer vision problems.

  Synthetic data that has a low SNR possibly approximates experimental data better than model data with high SNR. However, the SNR of the source task did not have any effect on generalization error. This suggests that for more effective (pre)training using model data, the human gait model must generate more varied data.

  The DCNN-2 needs relatively few parameter updates to reach good performance after pretraining by transfer learning, but the final performance is in all cases slightly worse compared with training the DCNN-2 from scratch. The transfer learning can be considered as a supervised counterpart to unsupervised pretraining such as was used with the AE in phase I, or as a special form of initialization. The effects observed here are opposite to that with the AE, as training speed and generalization error are increased. This shows that initialization of a neural network, even in case of normal learning behavior and convergence, can both positively and negatively affect training speed, training error and generalization error. These findings support the observation of [2] (p. 301) that initialization remains a poorly understood aspect of deep learning.

The transfer learning in this work was performed by re-initializing the final softmax classification layer and keeping the rest of the network. Different results might be obtained by only transferring the convolutional layers of the network and not the fully-connected layers.

Miscellaneous

- *Regularization.*

  The regularization applied to the various models proved to be reasonably effective, considering the modest amounts of overfitting observed in the experiments. The networks of phase I suffered the most overfitting, but these were also regularized the least.

  Regularizing by weight decay (L^2-norm) is effective, but it is likely that the optimal amount depends on the size and position of the layer in the network. This has not been investigated, however. [2] provides a general analysis of the effects of weight decay, but determining the optimal weight decay seems to be a matter of trial and error.

  Regularization by L^1-norm was not used in this study and its use seems to have fallen out of favor in modern deep CNN models. Its effect is by inducing sparsity of weights and hence sparsity of activations. It is known that the ReLU activation function promotes sparsity of activations as well, so the positive effects of L^1-norm regularization (if any) might be obtained by the use of the ReLU function.

  Dropout slowed down the training, explaining the remarkable observation that training error was often larger than the generalization error during initial stages of training, but strongly improved the generalization error.

  The rather large variance in test performance in the overfitting regime of the DCNN in the experimental multiple human gait scenario might be specific to the data since comparable results have not been found in literature, but it demonstrates the need for more regularization. Improving the network by more regularization is quite feasible since there are many regularization methods available and only a few of them have been investigated. Possible methods include batch normalization [51], clipping of gradients to prevent large parameters updates [52], injecting noise at the weights [53], at the gradients [54] or at the output targets (label smoothing) [55].

- *Visualization* The saliency map visualization technique provided some insight into the behavior of the DCNN, showing which parts of the spectrograms were generally important for the classification. However, in the case of wrongly classified samples, it was not distinctive enough to make clear why the DCNN had difficulty with a particular sample.

Limitations of this study.

It is important to note that the results obtained depend on the model parameters, which can have various interactions, and on the data used. The sheer amount of possible model configurations makes it unfeasible to exhaustively investigate the possibilities and control for possible interactions between variables.

A few issues with the data have been identified that may have had a large impact on the results. Firstly, by design of this study, it was decided to use only frontal aspect angles due to feasibility constraints. Using a range of aspect angles would undoubtedly lead to a much more difficult classification problem considering the large dependence of the micro-Doppler signature on aspect angle.

Secondly, the specific format and preprocessing of the spectrograms. A trade-off was made between size, computational complexity and human intelligibility of the spectrograms in determining its exact format. Also, the normalization of the spectrograms might have lead to overemphasizing the noise. Using larger spectrograms or other preprocessing might improve results.

Thirdly, the overlap between the classes of the experimental multiple human gaits. Manual inspection of the data revealed that at least a few dozens of samples belonging to the two-gaits or three-gaits classes could not be conclusively identified by a human observer. Other samples known to be produced by three humans might appear visually as a single or two-gaits class. Since these samples were legitimate data points, the data was not pruned of these and similar instances, which presumably would have lead to lower classification errors.

6

Conclusions

The main aim of this study was to investigate the merit of using Deep Learning techniques for human gait classification in radar. In this chapter, we summarize the main results in section 6.1 and state our final conclusions in section 6.2. The contributions of this work are summarized in section 6.3 and we give our recommendations for future work in section 6.4.

6.1 Research results

1. *What type of neural network is most suitable for the classification using spectrograms?*

   The convolutional neural network architecture is identified as the best architecture considered. The CNN based models are superior in generalization error and robustness against overfitting, while having less trainable parameters. CNNs need more computational resources and train slower than the MLP and AE architectures.

2. *What is the influence of the numbers of layers and neurons per layer (depth and width of the network)?*

   The effects of model size, both in width and depth, are rather limited for the MLP and AE architectures. Specifically, for the MLP increasing depth and width generally has small negative effects on stability and generalization error. For the AE, smaller networks perform slightly better while deeper networks suffer less from overfitting and train much slower.

   Deeper CNN models show better generalization error and less overfitting. The increase in depth also results in fewer trainable parameters and slower training.

3. *What is the influence of hyperparameters such as learning rate, optimization algorithms, parameter initialization and regularization methods?*

   Note that it is difficult to isolate the influence of each hyperparameter due to their interactions with one another and their dependence on both network model and data.

The learning rate must be set appropriately to let the networks converge and obtain optimal training speed, but in case of convergence, the effect on generalization error is very minor.

Using more advanced optimization algorithms can have dramatic positive influence on training speed while only slightly positively affecting generalization error.

The parameter initialization strongly affects the training performance, and positive effects on training dynamics may be accompanied by both positive and negative effects on generalization error. Parameter initialization remains poorly understood.

Regularization by penalizing the L^2-norm of the weights and Dropout both slow down training, but are crucial for obtaining low generalization error.

4. *What is the operational performance of the classifier in terms of classification accuracy, noise robustness, reliability, interpretability, training requirements and computational complexity?*

   The DCNN obtained a top accuracy of 85% percent, and when combining predictions of multiple samples and rejecting low-confidence samples this was increased to 90% on the experimental multi-target human gait classifications scenario. Despite the overall high accuracy of the network, it could make wrong predictions with high confidence.

   The DCNN showed good noise robustness in both the synthetic and experimental classification problems. For the experimental data, there was large variation in SNR though exact figures not are available.

   The visualization of the DCNN using saliency maps provided some insight into its inner workings, revealing intuitive behavior and supporting the general robustness of the network to noise. However, interpreting a deep neural network remains difficult.

   Training the DCNN took 3-12 hours using a high performance GPU (about 5 teraflops), depending mostly on the amount of training data used in the experiment. Classification of a single spectrogram takes less than a millisecond so real-time application is feasible.

   For application to experimental data, training a DCNN with synthetic data only does not suffice.

5. *How is the performance of the deep neural network classifier compared with other classification techniques, such as the 3-Nearest Neighbor classifier?*

   The DCNN achieved 85-90% accuracy, strongly outperforming the 3-NN which obtained an accuracy of 72%. The performance of both classifiers was very good for the non-gait and single gait classes, but the DCNN was much more accurate on the two-gaits and three-gaits classes.

6.2 Main conclusions

The main objective of this research was to assess the merit of using Deep Learning techniques for the purpose of human gait classification in radar. To this end, various Deep Learning based models have been applied to human gait classification problems.

Specifically, it was determined that the convolutional neural network architecture was superior to the MLP and AE architectures in classifying human walking versus human running versus a background class using micro-Doppler spectrograms. Subsequently, a deep convolutional neural network was designed and applied to a new, challenging multi-target human gait classification problem in which the number of human gaits was determined. The DCNN achieved high accuracy on synthetic and experimental data while strongly outperforming a 3-NN classifier in both cases. These results clearly demonstrate the merit of the Deep Learning approach in this particular domain.

Important advantages of Deep Learning techniques include the lack of need for manual feature engineering and extraction, and the ability to handle almost any type of data.

Potential difficulties and disadvantages include limited interpretability of the networks, the need for large amounts of training data and high computational cost of training the networks.

6.3 Contributions

The results and findings in this study make several contributions to literature:

- A deep convolutional neural network of 14 layers was used successfully for human gait classification. This is to our best knowledge by far the deepest neural network employed in the radar domain.
- It is possible to distinguish the number of walking people using micro-Doppler signatures for at least up to three persons. Related work only considered one versus two persons or one versus a group of persons.
- This work evaluated the effectiveness of training the DCNN with model data for application to measurement data.
- This work evaluated the performance of the DCNN for varying amounts of training data.

6.4 Future work

The findings presented in this thesis have raised several issues that provide a basis for further research. Significant limitations of this study were the format of the data and the aspect angles at which the targets were observed. Also, only a few classes were considered. Future work should extend the DCNN approach to more various targets that

are observed under various aspect angles, since the aspect angle has a large influence on the observed micro-Doppler signature.

Improving the data format by considering high resolution alternatives to the STFT such as the Wigner-Ville time frequency distribution might also be a worthwhile avenue for research.

Another promising topic would be the use of UWB radar and using a joint range-time-frequency representation which can be regarded as a sort of video-stream for which the DCNN approach seems well suited. This could even be extended to include the use of multi-static radar as well, as the DCNN can easily combine the multiple data sources.

In the Deep Learning domain, there are many open issues. One of the most pressing issues is the limited interpretability of neural networks. More advanced visualization methods are needed. These will also shed more light on the reliability of the networks.

The need of deep neural networks for large training sets hinders their application in domains where acquiring these amounts of data is infeasible, while the abstract features that these models provide can be useful in many applications. In the short term, more sophisticated data augmentation schemes and regularization methods may be developed. In the long term, the development of a neural network that is able to generalize from only a few examples, comparable to the learning of humans, will be an extremely challenging and rewarding research topic.

Both literature and the results of this study indicate that successful use of current neural networks comes down to using the model capacity effectively rather than just having large model capacity. With real-time applications in mind, it can be interesting to train small networks to mimic a large network. These compressed models may have (nearly) the same accuracy as the original network, but are of much smaller size and hence can be used for faster classification.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. Insight.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[3] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[7] Tom Le Paine, Pooya Khorrami, Wei Han, and Thomas S. Huang. An analysis of unsupervised pre-training in light of recent advances. *CoRR*, abs/1412.6597, 2014.

[8] Michael A. Nielsen. Neural networks and deep learning, 2015.

[9] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[11] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[13] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[14] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[16] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *CoRR*, abs/1301.3557, 2013.

[17] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[18] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[19] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010.

[20] Ronan Boulic, Nadia Magnenat-Thalmann, and Daniel Thalmann. A global human walking model with real-time kinematic personification. *The Visual Computer*, 6(6):344–358, 1990.

[21] Victor C Chen. *The micro-Doppler effect in radar*. Artech House, 2011.

[22] P. van Dorp and F.C.A. Groen. Human walking estimation with radar. *IEE Proceedings - Radar, Sonar and Navigation*, 150:356–365(9), October 2003.

[23] P. van Dorp and F.C.A. Groen. Feature-based human motion parameter estimation with radar. *Radar, Sonar & Navigation, IET*, 2(2):135–145, 2008.

[24] M.B. Guldogan, F. Gustafsson, U. Orguner, S. Björklund, H. Petersson, and A. Nezirovic. Human gait parameter estimation based on micro-doppler signatures using particle filters. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5940–5943, May 2011.

[25] Stephan Groot, Ronny Harmanny, Hans Driessen, and Alexander Yarovoy. Human motion classification using a particle filter approach: multiple model particle filtering applied to the micro-doppler spectrum. *International Journal of Microwave and Wireless Technologies*, 5:391–399, 6 2013.

[26] David Tahmoush Victor C. Chen and William J. Miceli, editors. *Radar micro-Doppler Signature. Processing and applications*. The Institution of Engineering and Technology, 2014.

[27] Youngwook Kim and Hao Ling. Human activity classification based on micro-doppler signatures using an artificial neural network. In *Antennas and Propagation Society International Symposium, 2008. AP-S 2008. IEEE*, pages 1–4, July 2008.

[28] Youngwook Kim and Hao Ling. Human activity classification based on micro-doppler signatures using a support vector machine. *Geoscience and Remote Sensing, IEEE Transactions on*, 47(5):1328–1337, May 2009.

[29] O.T. Alemdaroglu, C. Candan, and S. Koc. The radar application of micro doppler features from human motions. In *Radar Conference (RadarCon), 2015 IEEE*, pages 0374–0379, May 2015.

[30] D.P. Fairchild and R.M. Narayanan. Classification of human motions using empirical mode decomposition of human micro-doppler signatures. *Radar, Sonar Navigation, IET*, 8(5):425–434, June 2014.

[31] Irena Orović, Srdjan Stanković, and Moeness Amin. A new approach for classification of human gait based on time-frequency feature representations. *Signal Process.*, 91(6):1448–1456, June 2011.

[32] Fok Hing Chi Tivive, Abdesselam Bouzerdoum, and Moeness G. Amin. A human gait classification method based on radar doppler spectrograms. *EURASIP J. Adv. Signal Process*, 2010:10:1–10:12, March 2010.

[33] B. Lyonnet, C. Ioana, and M.G. Amin. Human gait classification using microdoppler time-frequency signal representations. In *Radar Conference, 2010 IEEE*, pages 915–919, May 2010.

[34] Jingli Li, Son Lam Phung, F.H.C. Tivive, and A. Bouzerdoum. Automatic classification of human motions using doppler radar. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6, June 2012.

[35] Youngwook Kim and Taesup Moon. Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 13(1):8–12, Jan 2016.

[36] Thayananthan Thayaparan, L Stanković, and I Djurović. Micro-doppler-based target detection and feature extraction in indoor and outdoor environments. *Journal of the Franklin Institute*, 345(6):700–722, 2008.

[37] Po Li, De-Chun Wang, and Lu Wang. Separation of micro-doppler signals based on time frequency filter and viterbi algorithm. *Signal, Image and Video Processing*, 7(3):593–605, 2013.

[38] Yazhou Wang and Aly E Fathy. Uwb micro-doppler radar for human gait analysis using joint range-time-frequency representation. In *SPIE Defense, Security, and Sensing*, pages 873404–873404. International Society for Optics and Photonics, 2013.

[39] Dave Tahmoush and Jerry Silvious. Gait variations in human micro-doppler. *International Journal of Electronics and Telecommunications*, 57(1):23–28, 2011.

[40] Francesco Fioranelli, Matthew Ritchie, and Hugh Griffiths. Aspect angle dependence and multistatic data fusion for micro-doppler classification of armed/unarmed personnel. *Radar, Sonar & Navigation, IET*, 9(9):1231–1239, 2015.

[41] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533, 2012.

[42] James Bergstra et al. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[43] Sander Dieleman et al. Lasagne: First release., August 2015.

[44] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, and S. Verzakov. Pr-tools4.1, a matlab toolbox for pattern recognition, 2007.

[45] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 249–256, May 2010.

[46] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[48] G. Urban, K. J. Geras, S. Ebrahimi Kaho, O. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, and M. Richardson. Do deep convolutional nets really need to be deep (or even convolutional)? *ArXiv e-prints*, mar 2016.

[49] Yann Dauphin and Yoshua Bengio. Big neural networks waste capacity. *CoRR*, abs/1301.3583, 2013.

[50] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. *arXiv preprint arXiv:1206.1106*, 2012.

[51] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[52] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[53] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[54] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[55] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.