CHALMERS
UNIVERSITY OF TECHNOLOGY

Radar Based Classification of Vulnerable Road Users

A comparison between two networks based on the ResNet and PointNet architectures and an evaluation of using time aggregated radar data for learned classifiers of vulnerable road users

Master's thesis in Systems, Control and Mechatronics

CHRISTIAN GARCIA
MÅNS LERJEFORS

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Radar Based Classification of Vulnerable Road Users

A comparison between two networks based on the ResNet and PointNet architectures and an evaluation of using time aggregated radar data for learned classifiers of vulnerable road users

CHRISTIAN GARCIA
MÅNS LERJEFORS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Radar Based Classification of Vulnerable Road Users
A comparison between two networks based on the ResNet and PointNet architectures and an evaluation of using time aggregated radar data for learned classifiers of vulnerable road users
CHRISTIAN GARCIA, MÅNS LERJEFORS

Supervisors: Christopher Zach, Chalmers University of Technology
Jianan Liu, Jeanette Warnborg, Alexander Lyckell, Aptiv Contract Services AB
Examiner: Christopher Zach, Electrical Engineering

Master's Thesis 2019
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualisation of how the findings of this thesis are intended to be used in traffic.

Typeset in LaTeX
Gothenburg, Sweden 2019

Radar Based Classifications of Vulnerable Road Users
A comparison between two networks based on the ResNet and PointNet architectures and an evaluation of using time aggregated radar data for learned classifiers of vulnerable road users
CHRISTIAN GARCIA, MÅNS LERJEFORS
Department of Electrical Engineering
Chalmers University of Technology

Abstract

As an increasing amount of automated features are integrated into vehicles today, there is a demand for a reliant system for detecting vulnerable road users. This thesis investigates the possibilities of classifying vulnerable road users based on solely radar data. It also explores the effect of using time aggregated data for different time spans. The investigation is done by comparing the performance of two different network architectures. One of the networks is inspired by the convolutional neural network ResNet and the other one by a neural network called PointNet which main application is to classify spatial point clouds. As input range-Doppler images and radar point clouds are used. The best performance is achieved by the ResNet-inspired architecture, with a time span ranging over three discrete data points. This achieves a accuracy of 92.59 %. The time aggregations of data is shown to have little to no effect in increasing the performance of either of the networks.

Keywords: deep neural networks, machine learning, radar, vulnerable road user classification, active safety.

Acknowledgements

We would like to thank the people that helped us during this thesis and made it possible. Thank you Mats Björnerbäck, first and foremost for giving us the opportunity to do this thesis at Aptiv and also for taking the time to discuss what type of thesis would be of use for Aptiv. Thank you Jonathan Jansson, Erik Larsson, Henric Eriksson and Jonas Lundberg for exchanging ideas and giving us feedback on our work. Thank you Jianan Liu for giving us an extensive introduction to the findings in machine learning that you found the most important for this thesis and for giving us a fundamental understanding of the work previously done at Aptiv. Thank you Jeanette for giving us valuable ideas when questions arose, for giving us feedback on our work and for helping us with technical issues. Thank you Alexander Lyckell, for providing the data and for explaining how Aptiv's radars work. And lastly, thank you Christopher Zach for taking the time and responsibility to be the examiner of this thesis and for providing us with ideas and feedback on how the work could be executed.

CHRISTIAN GARCIA, MÅNS LERJEFORS, Gothenburg, June 2019

Contents

List of Figures

List of Tables

List of Abbreviations and Nomenclature

Adam Adaptive moment estimation
CFAR Constant False Alarm Rate
CNN Convolutional Neural Network
Euro NCAP European New Car Assessment Programme
GPU Graphics Processing Unit
LFMCW Linear Frequency Modulated Continuous Wave
MLP Multilayer Perceptron
MSE Mean Squared Error
pp percentage points
ReLu Rectified Linear unit
resblock Residual building block
RMSprop Root Mean Square propagation
SISO Single-Input Single-Output
VRU Vulnerable Road User

Nomenclature

a Distance between two antennas
A Accuracy
AUC Area Under the Curve
b Any integer
B Bandwidth
c Speed of light in air
EPV Events Per Variable
f Received radar frequency
$\mathcal{F}$ Mapping of stacked nonlinear layers in a resblock
f_0 Emitted radar frequency
$F_{1,\mu}$ Micro average of F_1-score
FN False negative
FP False positive
FPR False positive rate
$f^*(\mathbf{x})$ True model

$h(\mathbf{x})$ Classifier
$\mathcal{H}$ Underlying mapping of a resblock
i Index for examples
I Time integrated range-Doppler map
j Index for classes
J Cost function
k Amount of iterations and splits in a k-fold cross-validation
K All time integrated range-Doppler maps from one logging stacked intro a struct
$\mathcal{L}$ Labelset
m Amount of points in a point cloud
M Logging with all range-Doppler maps
m_0 1st moment vector in the Adam optimisation algorithm
MR Exact Match Ratio
n Amount of data points in a dataset
$n_{\mathbf{B}}$ Batch size
$n_{\mathbf{D}}$ Doppler resolution of the radar
$n_{\mathbf{r}}$ Range resolution of the radar
p Amount of dimensions of one detection in a point cloud
P Precision
$p_{\mathbf{dropout}}$ Dropout parameter
$p_{\mathbf{network}}$ Amount of parameters in a network
q Amount of classes
r Range
R Recall
$RL(\mathbf{x})$ ReLu function
s Segment length
t Time
T Dataset
TN True negative
TP True positive
v_0 2nd moment vector in the Adam optimisation algorithm
v_r Velocity of the receiver
v_t Velocity of the target
w Learnable weight
$\mathbf{x}$ Point cloud, time integrated range-Doppler map, or a function input
X Batch
$\bar{\mathbf{x}}$ Normalised input $\mathbf{x}$
x Coordinate in x-dimension of a time integrated range-Doppler
$\mathbf{y}$ Function output
$\mathbf{Y}$ Labels belonging to example $\mathbf{x}$
y Coordinate in y-dimension of a time integrated range-Doppler
Z Output from classification function
α Stepsize
β Learnable bias term
z Coordinate in z-dimension
Δf Doppler shift

$\Delta\phi$ Difference in phase between antennas
Δt Integration time for a time aggregated data point
Γ Learnable parameters
γ_1 Learnable scaling factor used in batch normalisation
γ_2 Learnable shift parameter used in batch normalisation
λ Wavelength of emitted frequency
$\mu_{\mathbf{B}}$ Mean of a batch
$\sigma^2_{\mathbf{B}}$ Standard deviation of a batch
$\sigma(\mathbf{x})$ Sigmoid function
θ Azimuth angle from the boresight of the host to the target
ξ Activation function
ζ Decay rate

1

Introduction

With an automotive industry that is moving towards autonomous driving, more and more automated features are integrated into cars. Adaptive cruise control, intelligent speed adaptation and emergency brake assist are examples of features of this kind that are already common in modern cars. The aim of these features is to enhance car safety and thereby decrease the amount of road related accidents, as well as to reduce energy consumption and increase comfort.

Pedestrians and bicyclists, also known as Vulnerable Road Users, VRUs, are common elements in traffic, especially in the landscape of bigger cities. Car accidents with VRUs are one of the most common accidents happening due to driver distraction or misjudgement [1]. Hence, it is an area where automated safety features have a large impact. In Sweden for example, approximately 2000 pedestrians are injured in traffic related accidents every year [2]. One way to prevent these kind of accidents from happening would be for cars to have a reliant classification system for pedestrians implemented. Today VRU classification is done mainly with computer vision. The drawback of this approach is that it is sensitive to harsh weather conditions and disturbances such as dirt on the lens. Classification has been proven possible with radar, but not to the same extent as image classification. The radar approach suffers for example from its low resolution which complicates the classification process.

Machine learning has been found to be applicable for a large set of problems with outstanding results in recent years. Fields such as image recognition, spam detection, medical diagnosis, financial analysis and predictive maintenance are just a few areas where machine learning has excelled.

1.1 Purpose

The purpose of this thesis is to investigate to what extent VRUs can be classified using radar data. A successful radar based classification system could be a good compliment to the vision based systems that are mainly used today. With two types of sensors the system could be more robust against poor vision conditions.

In recent years automated safety features has been included as requirements when

the vehicle safety organisation Euro NCAP rates the safety of a car. In 2020 an auto emergency braking system that specifically reacts to VRUs will be incorporated in the test procedure [1]. Reliant classifications of VRUs is a way to make this feature possible.

1.2 Objective

The thesis investigates the radar based classification possibilities by comparing two network architectures. One of the networks is a convolutional neural network, CNN, based on the architecture of ResNet [3]. The other network is based on Multi Layer Perceptrons, MLPs, and is inspired by PointNet [4]. The two networks are developed to classify whether the radar data contains a *car*, *pedestrian*, and/or a *bicyclist*. The thesis is done at the company Aptiv Contract Services AB, in their office in Gothenburg, Sweden.

Two datasets are created where one of them contains driving scenarios inspired by driving scenarios defined by Euro NCAP. The driving scenarios used for the two datasets are defined in A.2 and A.3. Each network is trained and tested on both datasets separately. The networks capability to generalise is studied by using one of the datasets as training set and the other dataset as testing set.

Due to the difference in architecture the two networks require radar data preprocessed in different ways. The CNN-based architecture is fed radar data in the form of 2D images, or maps, which display the received radar signal amplitude over the dimensions range and Doppler shift. The MLP-based network, inspired by the PointNet [4], is fed radar data on the form of a data point cloud. The data is preprocessed and time aggregated for an evaluation on whether this can facilitate the classification of the networks. Hence the evaluation is a comparison between the two network architectures and an investigation of the feasibility of the two radar preprocessing methods. The radar data will also be filtered to evaluate to which extent this has an effect on the networks performance.

1.3 Scope

This thesis is limited to only perform classifications based on the total radar input. Neither of the two networks are able to give any information regarding where the detected object is located, but instead only determine whether the radar data contains a *car*, *pedestrian*, and/or *bicyclist* or not. This is to avoid the extensive amount of manual labelling it would require.

The evaluation is based on data provided by the company Aptiv and is taken from a logging session done to test their products. The scenarios in these logging session

are partly inspired by the Euro NCAP scenarios for VRU detection. The evaluation is hence a proof of concept and not a direct evaluation of the feasibility of a direct implementation of the results from this study in real world scenarios.

1.4 Scientific contribution

This thesis aims to make a contribution in the area of radar based classification using machine learning. Its main contributions are:

- A comparison between the CNN-based architecture and the PointNet inspired architecture and their respective radar data input.
- An evaluation of how time aggregating radar data affects the networks performance.
- An investigation of how filtered radar data affects the networks performance.

1.5 Outline of thesis

Except from the brief introduction to the problem introduced in **Chapter 1**, this thesis consists of five additional chapters.

Chapter 2 serves as a background chapter providing the reader with the basic knowledge within the field. It partly consists of an explanatory section describing the key concepts of radar technology and an overview of how radar data is commonly illustrated. It also consists of a section containing the fundamentals behind neural networks, an introduction to the two network architectures used in this thesis and describe commonly used evaluation metrics for this kind of problem. The chapter ends with an overview of related work in the research area.

Chapter 3 covers the methods used in this thesis. It explains the datasets used to train and validate the networks and how these datasets are retrieved and preprocessed. It also thoroughly describes the networks used in this thesis.

Chapter 4 consists of the results gathered by comparing the two networks on the two datasets. These results are then discussed in **Chapter 5** and the final conclusion of the thesis can be read in **Chapter 6**.

2

Background

This section will go through the necessary theory in order to get an understanding of the key concepts covered in this thesis. It begins by explaining the basics of a radar and the specific algorithms and concepts behind the radar used in this thesis. It continues with an introduction to the theory behind neural networks and an explanation of the used network architectures. This section also brings up by what metrics the networks are evaluated, and how the training is done to make sure that the result of the performance is portrayed fairly. Lastly, related work in the scientific area is discussed.

2.1 Radar

The primary usage of a radar is to determine the characteristics of the surrounding environment based on how a transmitted electromagnetic wave is reflected back. A basic radar set up is composed of two components, a transmitter and a receiver. A signal that is reflected back to the receiver is denoted as a detection. One transmitted signal can cause many detections. These detections can be caused by both the ground and surrounding objects. In radar terminology an object of interest is often denoted as a target. A target can yield several radar detections as is illustrated in Figure 2.1. A radar can determine the distance to a detection and hence also a target by using the received arrival time of the transmitted signal. The range, r, to a detection can then be computed by the fairly simple equation,

$$r = \frac{c\,t}{2}, \tag{2.1}$$

where t corresponds to the time it takes for the signal to echo back to the radar and c corresponds to the velocity of waves in the medium, which in this case is the speed of light in air.

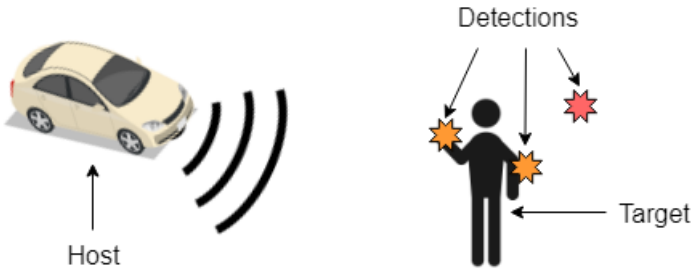


Figure 2.1: Illustration of a host vehicle with a radar mounted in the front. The radar yields three detections, where two detections belongs to a target, which in this case is a pedestrian. The detections of interest are orange.

Besides from range, a radar have the possibility to measure the velocity by making use of the Doppler effect. This phenomenon is described by the equation for Doppler shift, Δf, which explains the difference in frequency between the emitted and received signal by

$$\Delta f = \frac{\Delta v}{c} f_0 = \Delta v \lambda, \tag{2.2}$$

where $\Delta f = f - f_0$, $\Delta v = v_r - v_t$, and v_r is the velocity of the receiver, v_t is the velocity of the target, f the received frequency, f_0 is the emitted frequency, and λ is the wavelength of the emitted frequency [5]. Hence, the measured speeds will be the relative radial velocity to the radar. This means that an object travelling in a circle around a radar will have a measured relative radial velocity of zero [6].

2.1.1 Frequency modulation

In order to compute the velocity of a target, a radar transmitting a continuous wave with a fixed frequency could be used. As stated above, the target velocity can then be computed with equation (2.2). However, with this type of radar it is not possible to compute the range to the target [6]. Due to this, several frequency modulation techniques has been developed in order to gain information about the distance to a target.

One of the most common techniques within the automotive industry is called linear frequency modulated continuous wave, LFMCW [6]. This is also the modulation technique used by the radars in this thesis. The principles of LFMCW is depicted in Figure 2.2.

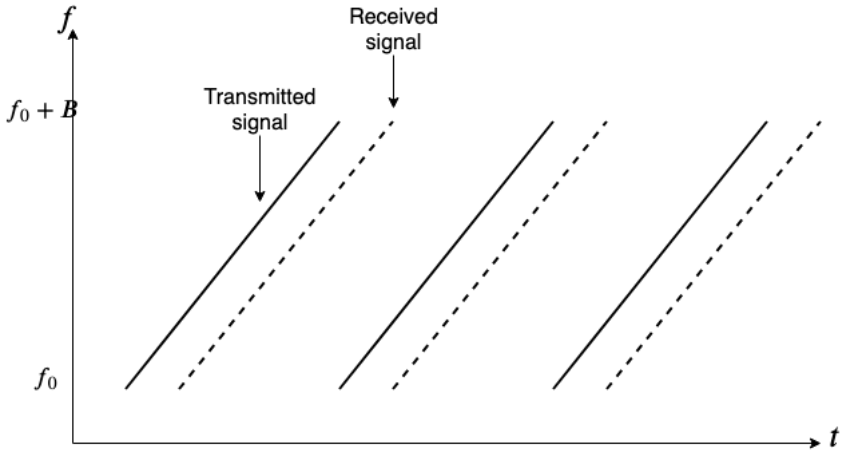


Figure 2.2: Illustration of the linear frequency modulation continuous wave technique with three chirps.

Instead of using a fixed frequency, as would have been done in a simple continuous wave radar, a LFMCW-radar lets the frequency vary from a minimum frequency f_0 to a frequency $f_0 + B$, where B corresponds to the bandwidth. A frequency sweep like this is referred to as a chirp. During a single measurement a LFMCW-radar transmits multiple chirps. By measuring the difference in frequency Δf, between the transmitted frequency and the received frequency, the range can be computed. This is possible due to the range being proportional to the the linear frequency change. If the target is moving, adjustments to the target induced frequency shift also has to be made. The radar used to gather data for this report acquires data at a rate of 20 Hz. In this thesis one single measurement instance will be referred to as a data point. One data point will hence be all detections gathered from the same measurement instance.

2.1.2 Azimuth angle

When a radar is equipped with multiple receiving antennas the angle of which the object is located is possible to compute. This can be done by measuring the difference in phase, $\Delta\phi$, between the receiving antennas. In automotive purposes the angle of interest is the angle in the horizontal plane. This angle is called the azimuth angle, θ, and can be computed by

$$\theta = \sin^{-1}\left(\frac{\lambda}{2\pi a}(\Delta\phi + 2\pi b)\right), \tag{2.3}$$

where λ is the wavelength, a is the distance between the two antennas used for the calculation, and b can be set to any integer to solve the equation since the sine function is periodic. With the azimuth angle it is possible to estimate, not only at

which distance the target is located, but also in which direction the target can be found. The complete algorithm to find the azimuth angle can be found in [7].

2.1.3 Constant false alarm rate

The signal detected by a radar receiver will consist of both noise caused by the internal components of the receiver and of noise caused by the surroundings. If the noise is low enough a simple solution to this problem would be to define a certain threshold for the signal strength and filter out all signals below the defined threshold. A low threshold would yield many false alarms but simultaneously not bear the risk of missing real targets, while a high threshold would yield few false alarms but would have a higher risk of filtering out real detections.

The set up of having a fixed threshold could work fairly well in a fixed environment with a stationary radar, but when the surroundings change it is hard to set a decent fixed threshold value. The purpose of the Constant False Alarm Rate, CFAR, algorithm is to let the threshold value vary and hence making it adaptable to new environments where the background noise, for instance, is higher. There are several different CFAR-algorithms that estimate the varying threshold value in different ways [8]. The specific CFAR-algorithm used for this study is confidential.

2.1.4 Micro-Doppler

A commonly used method to visualise radar data is through micro-Doppler images. These images depict the micro-Doppler effect, which is a phenomenon occurring when an object has multiple detection points with diferent speeds relative the radar and thus reflects back different Doppler frequencies. For instance, a walking human would yield a range of different Doppler speeds. The signals reflected off the torso would correspond to the speed in which the person is heading, while the arms and legs are travelling in other relative speeds. Over time, this yields characteristic patterns called micro-Doppler signatures, which vary depending on the studied object. E.g. a car would not have the pattern that a pair of swinging arms cause in its micro-Doppler signature, since all parts of the car travels at the same speed. Even though a car passing at close distance to the radar would have both positive and negative speeds while being directly in front of the radar, since one part of the car is travelling away from the radar and the other part is travelling towards the radar. The micro-Doppler map from a car can be seen in Figure 2.3. A micro-Doppler map normally has frequency on the y-axis and time on the x-axis. The intensity for each x- and y-value is then normally plotted as a heat map on the surface spanned by x and y.

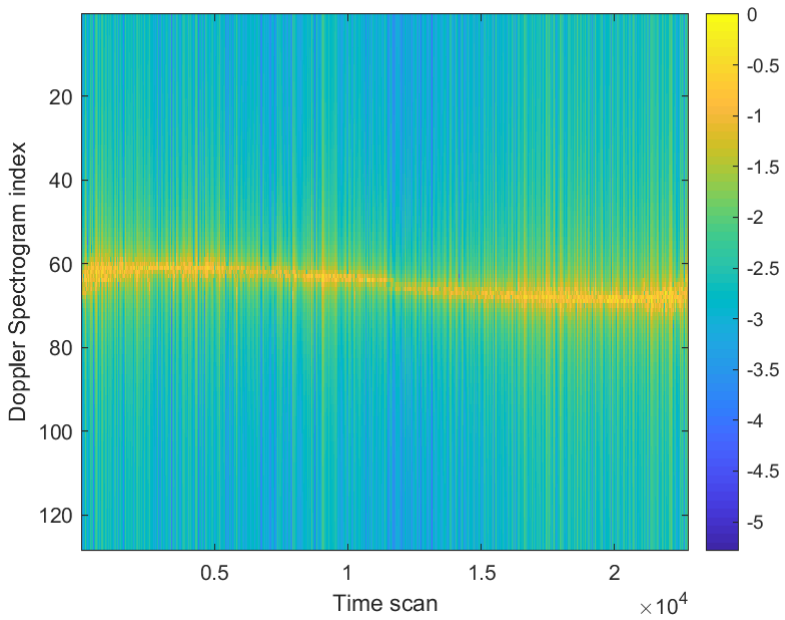


Figure 2.3: Micro-Doppler map. The measurement comes from a car driving in a circle in front of a radar for approximately 30 seconds.

2.1.5 Time integrated range-Doppler

Range-Doppler maps are another common way to display radar data. In a range-Doppler map, the computed range is depicted against the computed Doppler velocity, also known as Doppler shift. The magnitude of the reflected signal is illustrated with colours, ranging from red to dark blue, where red represents the largest reflected values and blue the lowest.

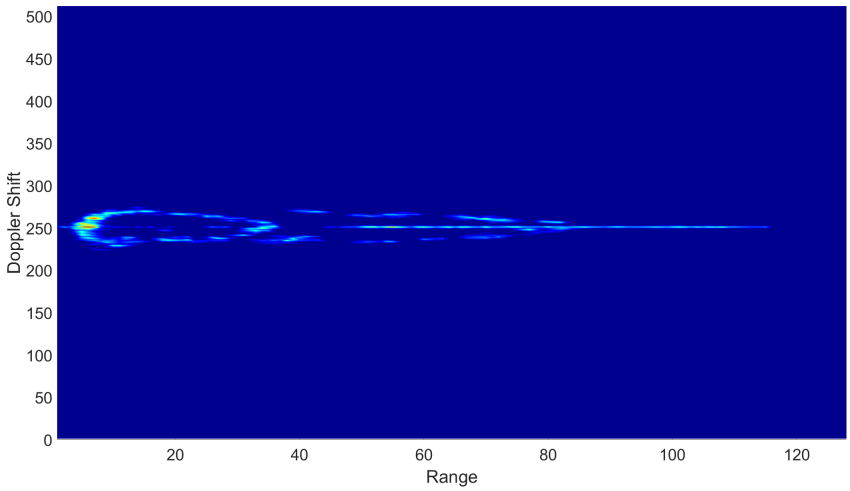


Figure 2.4: Integrated range-Doppler map. The measurement comes from a man riding a bicycle in a circle in front of a radar approximately 30 seconds.

In [9] a way to combine the spectrogram-like features of a micro-Doppler with the range data in range-Doppler maps is proposed. The proposed approach is time integrated range-Doppler maps. By taking the maximum pixel value over a time span, Δt, an objects time-correlated features can be visualised and extracted. An example of time integrated range-Doppler map can be seen in Figure 2.4.

2.2 Neural networks

Deep neural networks are a modern subcategory of machine learning. The naming is derived from it being loosely inspired of the neurons in the human brain. These type of networks are of high importance in many applications today, such as computer vision and natural language processing.

The fundamental architecture of deep neural networks is based on the fully connected layer. This layer contains neurons where each neuron is connected to all neurons in the adjacent layers and not connected to any neurons in the same layer. A neuron in a neural network is simply a unit that takes several inputs and computes an activation value to pass forward to neurons in the next layer. The overall goal of the network is to approximate the true model $f^*(\mathbf{x})$, by the network model $h(\mathbf{x})$, based on the input $\mathbf{x}$. An illustration of a simple fully connected network can be seen in Figure 2.5. An architecture based on these layers is generally known as a multilayer perceptron or MLP.

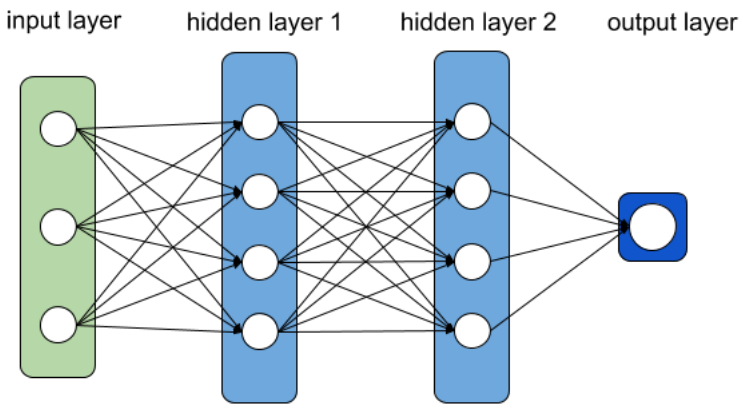


Figure 2.5: A conventional fully connected neural network with three layers, three inputs, four neurons per layer and one output.

2.2.1 Activation function

The activation function is mainly used to map the output values from a layer to suitable values that will serve as input to the neurons in the next layer. The activa-

tion function introduces nonlinear properties to the neural network. Two commonly used activation functions are the sigmoid function, $\sigma(\mathbf{x})$, and the rectified linear unit, ReLu, function, $RL(\mathbf{x})$. The functions are explained by

$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}}, \tag{2.4a}$$

$$RL(\mathbf{x}) = \max(0, \mathbf{x}), \tag{2.4b}$$

respectively, and are presented visually in Figure 2.6.

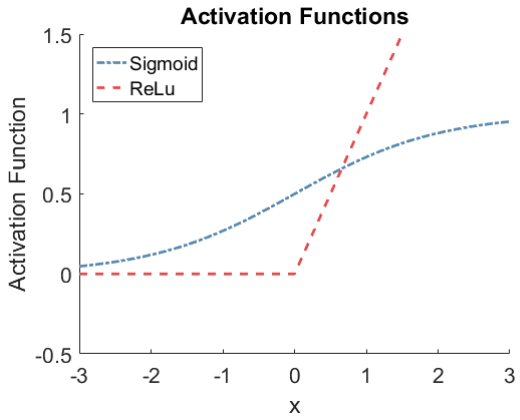


Figure 2.6: The figure illustrates the sigmoid function and the ReLu function explained in equations (2.4a) and (2.4b) respectively.

When dealing with classifiers it is preferable to have an activation function in the final layer of the network that yields a probabilistic output. The sigmoid function does exactly this. However, the univariate sigmoid function is only applicable in binary classification cases, since it gives a probability of a statement being either true or false. This is since the sigmoid function will map the final output of a neural network to a single probabilistic value ranging from 0 to 1. In other terms, the probability that the input belongs to a class or not. In cases with multiple classes there are a few different approaches to solve the classification problem. One approach is to use multi-class classification, and let each input only be classified as belonging to one class. Another approach is to use multiple binary classifiers, which instead defines a problem as a multi-label classification problem. In this case, multiple sigmoid functions, one per label, could be used as a final layer.

2.2.2 Learning, backpropagation and loss

In order to properly estimate the true model, $f^*(\mathbf{x})$, the network has learnable parameters. Each neuron has learnable weights, w_i, and a learnable bias term, β. The input, $\mathbf{x}_i$, is multiplied with a corresponding weight, w_i, which is then added with the bias, β. This is done for every input to the neuron and then summed together. The resulting value of the summation is put through an activation function, ξ, which

then constitutes the output $\mathbf{y}$ of the neuron. Hence, the output can be expressed as $y = \xi(w^T\mathbf{x} + \beta)$. This output then serves as input to the neurons in the next layer. The complete computational flow of a single neuron is illustrated in Figure 2.7.

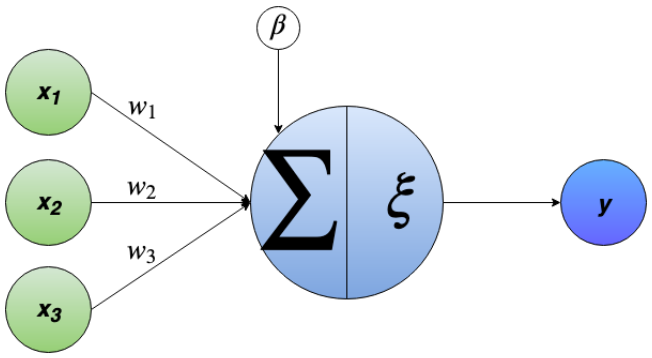


Figure 2.7: The computational flow of a neuron, with three inputs and a bias term.

When the network is trained, an input $\mathbf{x}$ is inserted to the network, which by letting the data flow through all layers of the network produces an output $\mathbf{Y}$. This process is called forward propagation. For each network a loss function is defined, which gives an estimate of how well the network performs. An example of a loss function is the mean squared error, MSE, where the loss, J, is defined as

$$J = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Z_i)^2 \tag{2.5}$$

where $\mathbf{Y}_i$ is the target variable, Z_i is the output predicted by the network and n is the number of samples being predicted. The loss can easily be computed after forward propagation. The choice of loss function is dependent on which type of application the network is designed for. MSE is one of the most commonly used loss functions for regression problems. One commonly used loss functions dealing with multiple binary classification problems is the Multi Label Soft Margin Loss [10], which is formulated below,

$$J = -\frac{1}{q}\sum_{i=1}^{n}\left(Y_i \log\left(\frac{e^{Z_i}}{1+e^{Z_i}}\right) + (1 - Y_i)\log\left(\frac{1}{1+e^{Z_i}}\right)\right), \tag{2.6}$$

where q corresponds to the number of labels.

To update the value of the learnable parameters, Γ, backpropagation is done. Backpropagation refers to the process of computing the gradient of the loss with respect to the parameters, $\nabla_\Gamma J(\Gamma)$. Hence, the weights and biases will be updated in a manner that produces a lower loss in the next forward propagation. The computations of the gradients in every layer are done with the chain rule.

In most deep learning applications the complete dataset is divided into batches. Large batch size are computationally faster, while small batch size has the advantage of bringing better generalisation performance. Both [11] and [12] conclude that a batch size of 32 is a good compromise. New parameter values are computed by doing backpropagation for every batch. An epoch is referring to when all batches of a dataset have been used to update the parameter values. The training of a network usually consists of several epochs of parameter updating and backpropagation [13].

2.2.3 Optimisation algorithm

There are several different optimisation algorithms. The optimisation algorithm is used to calculate how the parameters of the solution, Γ, are going to be updated. This is done by taking a step of length α in the direction of the gradient calculated by the algorithm. The step size is often referred to as learning rate. One of the most popular algorithms is Adam, which uses the benefits of momentum and root mean square propagation, RMSprop [14]. Momentum pushes the solution in the direction of the previous gradient and thus creating "momentum" and RMSprop makes the method take smaller steps in steep directions and bigger steps in less steep directions [14]. The algorithm described by

Algorithm 1

The stochastic optimisation algorithm, Adam, is best initialised with the stepsize $\alpha = 0.001$, $\epsilon = 10^{-8}$, $\zeta_1 = 0.9$, and $\zeta_2 = 0.999$ [14]. All vector operations are applied element-wise. ζ_1 and ζ_2 to the power of t is denoted ζ_1^t and ζ_2^t.

Require: α: Stepsize
Require: $\zeta_1, \zeta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\Gamma)$: Stochastic objective function with parameters Γ
Require: Γ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialise 1st moment vector)
$v_0 \leftarrow 0$ (Initialise 2nd moment vector)
$t \leftarrow 0$ (Initialise timestep)
while Γ_t *not converged* **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\Gamma f_t(\Gamma_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
  $m_t \leftarrow \zeta_1 \cdot m_{t-1} + (1 - \zeta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \zeta_2 \cdot v_{t-1} + (1 - \zeta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_t \leftarrow \frac{m_t}{1-\zeta_1^t}$ (Compute bias-corrected first moment estimate)
  $\hat{v}_t \leftarrow \frac{v_t}{1-\zeta_2^t}$ (Compute bias-corrected second raw moment estimate)
  $\Gamma_t \leftarrow \Gamma_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ (Update parameters)
end
return Γ_t (Resulting parameters)

.

The benefits of Adam is it being computational efficient, requires small memory usage and is suitable for large datasets [14].

2.2.4 Convolutional layer

A convolutional neural network, CNN, is a specific kind of neural network. The architecture has proved to be extremely efficient in image recognition related tasks. A key component of the CNNs is the convolutional filters.

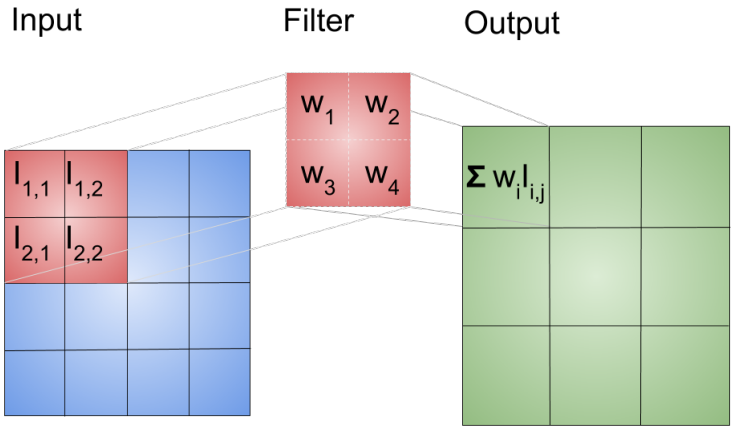


Figure 2.8: A convolutional filter acting on an input image. In the figure the convolutional filter acts on three image patches per row and three image patches per column. Thereof the output is a three-by-three matrix.

The convolutional filter is an, often square, matrix of learnable weights. The dot product is performed between the weights in the filter and an equally sized patch in the input image. This product then becomes the value of the element in the corresponding place of the output. The stride of a convolutional filter is the amount of steps in pixels the filter is ”moved” before acting on the next input patch. In Figure 2.8 a stride of one is used. The filter is applied from left to right and from top to bottom. The output of a convolutional filter is called activation map.

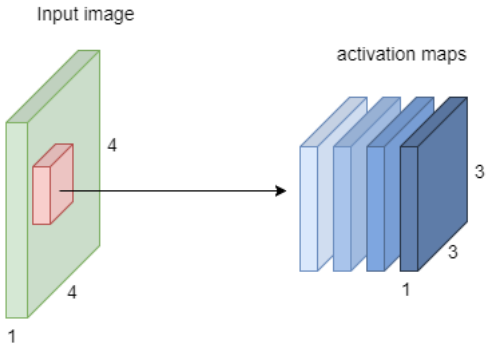


Figure 2.9: Illustration of four 3×3×1 activation maps yielded by four 2×2×1 filters.

A convolutional layer usually consists of several convolutional filters, resulting in several activation maps. This yields an output with a depth corresponding to the amount of filters used in the convolutional layer. An illustration of a convolutional layer, consisting of the same input image and filter size as used in Figure 2.8, can be seen in Figure 2.9.

2.2.5 Overfitting and dropout

For a neural network it is important to perform well on previously unseen data. This ability is called generalisation in machine learning vocabulary. If the network is overfitting it does not generalise well, which means that there is a large difference in network performance when it is tested on training data and when it is tested on new data. There are several reasons to why overfitting will occur. One often mentioned reason is having more network parameters than training samples in the dataset [15].

There is, however, many ways to reduce the risk of overfitting. Dropout is one commonly used method to do exactly this. The concept behind dropout is fairly simple. For each step in the training phase a random fraction of neurons, p_{dropout}, will be dropped out, i.e. ignored. This means that in a case where the dropout rate is set to $p_{\text{dropout}} = 0.5$, half of the neurons will be randomly ignored throughout the training process. This is done in order to avoid a network that is very dependant on a few neurons for making a proper classification. With dropout all neurons is forced to learn something about the data. This significantly reduces the risk of overfitting [16].

2.2.6 Batch normalisation

Every time the weights are updated the distribution of a hidden layer's input is changed. This requires the network to have a low learning rate, which slows down

the learning [17]. Batch normalisation refers to the procedure of normalising the input to a succeeding hidden layer in order to solve this problem. The normalisation is done for every batch, $X = \{\mathbf{x}_1, ..., \mathbf{x}_{n_\mathrm{B}}\}$, where n_B denotes the batch size. The normalisation scheme is described below

$$\bar{\mathbf{x}}_i = \frac{\mathbf{x}_i - \mu_\mathrm{B}}{\sqrt{\sigma_\mathrm{B}^2 + \epsilon}}, \tag{2.7a}$$

$$\mathbf{y}_i = \gamma_1 \bar{\mathbf{x}}_i + \gamma_2, \tag{2.7b}$$

where μ_B and σ_B^2 corresponds to the mean and variance of the batch being considered. γ_1 is a learnable scaling factor and γ_2 is a learnable shift parameter. ϵ is a small number introduced to avoid division by zero. Hence, $\bar{\mathbf{x}}$ is the normalisation of the input $\mathbf{x}_i$ and $\mathbf{y}_i$ the output from the batch normalisation. This process is done in every neuron. Batch normalisation is shown to not only speed up the learning process, but also to reduce the risk of overfitting [17].

2.2.7 Resblock

Even though a networks ability to generalise increases with the depth of the network, beyond a certain depth adding layers to a network can lead to that the accuracy stagnates or even degrades [18]. This is partially due to the vanishing gradient problem [3][19][20]. A widely used approach to combat this issue is the use of residual building blocks, or resblocks, from [3] where the ResNet architecture is explained. The idea behind the ResNet is to not guess that the stacked layers directly fit an underlying mapping, but instead let the layers explicitly fit a residual mapping. To do this the underlying mapping is defined as $\mathcal{H}(\mathbf{x})$ and the stacked nonlinear layers fit the mapping $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - x$. The idea is that if an identity mapping is optimal or at least a close enough mapping, then it is easier to get the residual to zero than to find an identity mapping with nonlinear layers.

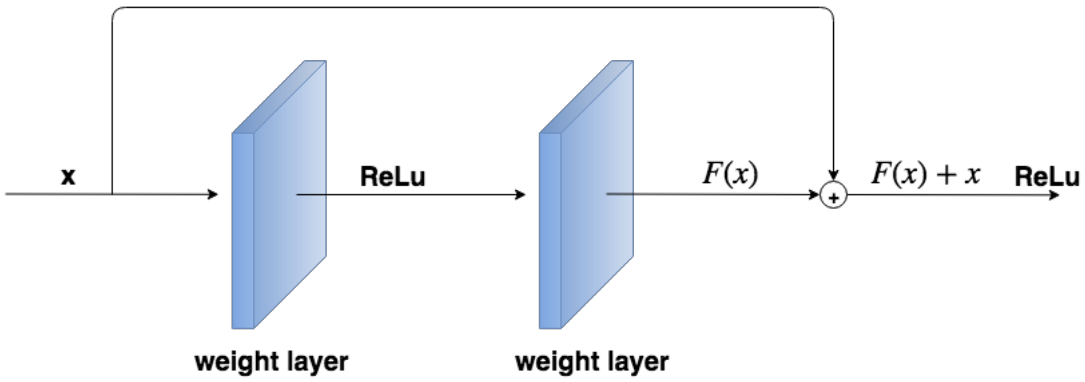


Figure 2.10: Visualisation of a resblock.

The identity mapping is realised by shortcut connections as illustrated in Figure 2.10.

Using resblocks as a building block for a network helps avoiding the vanishing gradient and exploding gradient problem [3].

2.2.8 PointNet

PointNet is a network designed to consume point cloud data and perform object classification and part segmentation on the dataset. This is desirable since point clouds resembles the way raw sensor data is received. In PointNet each point is processed independently. In the basic architecture a point is represented by 3D coordinates (x, y,z). Additional dimensions, e.g. colour and normal, can be added [4].

In order to successfully classify point clouds two main challenges are solved by PointNet. The first one is the problem of being invariant to the order of which the points are fed to the network. The solution proposed in PointNet is a structure with a shared MLP for all points, followed by a max pooling and an MLP. The max pooling acts as a symmetric function and is hence making PointNet invariant to permutations. An illustration of this implementation can be seen in Figure 2.11. The second problem solved by PointNet is the problem of being invariant to point cloud rotations. By letting a small version of the network, called T-net, predict an affine transformation matrix PointNet is able to align the input points. This module is illustrated in Figure 2.12.

The PointNet architecture has been proven successful of performing part segmentation and classification on radar point clouds in [21]. This approach uses two spatial coordinates, (x, y), and two additional dimensions which can be found in [21].

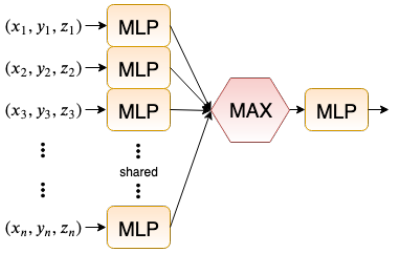


Figure 2.11: Architecture of the point order invariance module with n number of points.The multiple rows of MLPs to the left illustrates the MLP is shared, i.e. it is the same MLP used for all points.

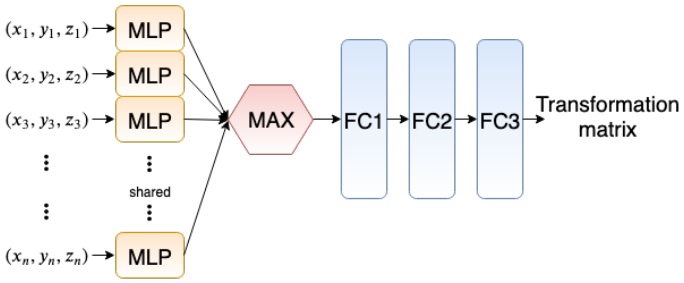


Figure 2.12: Architecture of the T-net module. The multiple rows of MLPs to the left illustrates that the it is the same MLP used for all points.

2.3 Classification problem and evaluation metrics

There are several ways to measure the performance of a network. Different metrics measures different aspects of the networks performance. Certain metrics are better suited for some classification problems than others. This section will explain how the problem is defined for the networks and the metrics being used to evaluate them.

2.3.1 Binary relevance problem

The classification problem in this thesis is defined as a binary relevance problem [22]. This approach trains one binary classifier for each label. The model independently predicts each label in one example. To do this a dataset needs to be defined.

A dataset, T, is defined by its n examples $(\mathbf{x}_i, \mathbf{Y}_i)$, $1 \leq i \leq n$. The examples are defined by $(\mathbf{x}_i \in \mathcal{X}, \mathbf{Y}_i \in \mathcal{Y} = \{0,1\}^q)$, where $\mathbf{x}_i$ is a the input to be classified, and $\mathbf{Y}_i$ contains the binary true labels associated with $\mathbf{x}_i$. The datasets include a labelset $\mathcal{L}$, where the labels $l_j \in \mathcal{L}$, $1 \leq j \leq q$, and $|\mathcal{L}| = q$. A classifier, h, classifies an example $\mathbf{x}_i$ by $h(\mathbf{x}_i)$. Each classification outputs q predicted labels, that is $h(\mathbf{x}_i) = Z_i = (z_1, .., z_q)$. Ideally $Z_i = \mathbf{Y}_i, \forall i$. This translates to the problem *"Does label l_j belong to $\mathbf{x}_i$?"*.

The general disadvantage with the binary relevance problem is that it does not model label dependency. This should not be a disadvantage for this particular classification problem since label dependency is not desirable. A case where label dependency would be of interest is for example when classifying movies. A movie is likely to be correctly labelled *family friendly* and *comedy* at the same time, but not *horror* and *family friendly*. For this particular classification problem the probability of the presence of a pedestrian should not be dependent on the probability of the presence of a car for example.

2.3.2 Metrics of networks

A common way to measure network performance is by computing the accuracy, A, which in this case where the problem is defined as a binary relevance problem, is defined by

$$A = \frac{\sum_{j=1}^{q} \left(\mathrm{TP}_j + \mathrm{TN}_j \right)}{\sum_{j=1}^{q} \left(\mathrm{TP}_j + \mathrm{FP}_j + \mathrm{TN}_j + \mathrm{FN}_j \right)}, \tag{2.8}$$

where T, F, P and N in TP, TN, FP and FN stand for true, false, positive and negative, and q is the number of labels. A true positive, TP, is a classified example that has been classified as label_j and does belong to label_j. TN, FP and FN are

in analogy with TP. Hence A is a measurement of how well a network is classifying overall, without giving any importance to a particular label.

If a network however is able to classify an example as containing multiple labels at once, accuracy does not paint the whole picture. The exact match ratio, MR, is a more strict version of accuracy where all predicted labels of an input must be correctly classified to contribute to the score. This metric gives a measure of the ratio between the amount of examples completely correctly classified to the amount of examples classified in total.

$$MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i), \tag{2.9}$$

where I is the indicator function, Z_i the predicted labels, Y_i the true labels and n the amount of examples being evaluated. MR does not take partially correct classifications in to consideration. Partially correct classifications are counted as incorrect classifications.

In binary classification precision and recall are two commonly used measures. The precision, P, is defined by

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{2.10}$$

P is a measurement of how precise a network is each time it classifies an object as containing a particular label. A high P value would suggest that the network, when classifying an object as containing a specific label, is most often correct. Recall, R is defined by

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{2.11}$$

R does instead give high values for a specific label if a high amount of the examples are classified as containing that specific label. The downside to this measurement is that a network that tends to over-classify objects as a specific class would give a high value of R. The harmonic mean of R and P is called F_1, and is defined by

$$F_1 = 2 \cdot \frac{R \cdot P}{R + P}. \tag{2.12}$$

F_1 is a measure of the balance between P and R, or the harmonic mean of P and R. In datasets where there is a relatively large imbalance of labels, it is better to use the micro average of F_1 than the macro average. The micro average $F_{1,\mu}$ is calculated by

$$F_{1,\mu} = 2 \cdot \frac{\sum_{j=1}^{q} \mathrm{TP}_j}{\sum_{j=1}^{q} \left(2\mathrm{TP}_j + \mathrm{FP}_j + \mathrm{FN}_j\right)} \tag{2.13}$$

where TP_j and FP_j are the amount of true positives and false positives for label l_j respectively, and q is the number of labels.

The false positive rate FPR, is a measure of how often a classifier wrongly classifies an example as positive label, when it actually is negative, per total amount of negative examples. It is defined by

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{2.14}$$

Another performance measure is the Precision-Recall curve in combination with its area under the curve, *AUC*. This curve is a plot of the precision on the vertical axis and the recall on the horizontal axis, for different thresholds in the last step of the classifier. As explained in Section 2.2.1 each binary classifier outputs a probabilistic output between 0 and 1. The thresholds values in question are the values for where a prediction is being considered true. Hence, a Precision-Recall curve displays how a binary classifier is affected by choosing different threshold values. AUC is a metric of how good the Precision-Recall curve is and is simply calculated as the area below the curve, with a maximum of 1.

A metric on how to evaluate the number of parameters in a network compared to the number of examples in the dataset is the events per variable, *EPV*, as suggested in [23] and [24] for regression models. The metric is defined by

$$EPV = \frac{n}{p_{\mathrm{network}}}, \tag{2.15}$$

where n is the number of examples in a given dataset and p_{network} is the number of parameters in a network.

2.3.3 *k*-fold cross-validation

Cross-validation is used to estimate the expected performance. It is also used to select the best fit model and to ensure that the model is not overfitting. The k-fold cross-validation method is implemented by splitting up the dataset in to test and training data k times. Each time the size of the test dataset is $\frac{1}{k}$:th of the full dataset. The k different test datasets are chosen so that no test set has overlapping data with another test set. The remaining data is the training data. The concept is visualised in Figure 2.13.

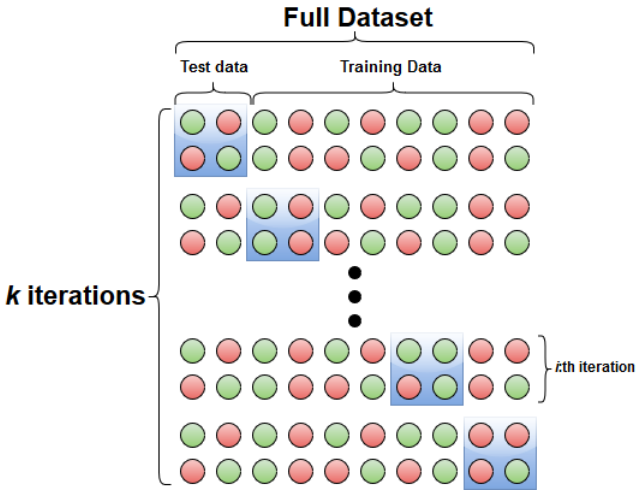


Figure 2.13: An illustration of how the test and training data is chosen between k iterations in k-fold cross-validation.

Which k to choose is a trade-off between choosing a large k and thus not perturbing the data enough, and a small k which leads to a small training set relative to the full dataset. Choosing $k = 5$ is considered a good compromise between the two [25].

2.4 Related work

The main area of active safety related detection and classification research has been conducted within vision based systems. In [26] a computer vision based system for real time vehicle tracking is proposed. The proposed system is proven to be robust against harsh conditions such as occlusion, varying lightning conditions, and vibrations. A system to perform vision based pedestrian detection on-board of a moving vehicle is presented in [27]. In [28] pedestrian classification based on a single frame is investigated with the conclusion that some features need to be measured over time in order to get reliant classifications.

In regards to image recognition alone, there has been done extensive research in developing highly effective network architectures in order to enhance the classification performance. The ResNet is presented in [3]. This network uses identity mapping to surpass the vanishing or exploding gradient problem. The ResNet allows a deep neural network architecture, which will be used in this work but with a smaller amount of parameters. The densely connected convolutional network, *DenseNet*, presented in [29], connects all layers to each other and achieves to substantially reduce the number of parameters and alleviate the vanishing gradient problem even further. Batch normalisation gives the benefit of achieving the same accuracy with substantially fewer training steps [17]. In [30] it is shown that under certain conditions and assumptions all bad local minima can be removed by adding a neuron. In [31] it is shown that this can be done for any neural network, for multi-class classification,

for binary classification, and regression with an arbitrary loss function.

Classification of VRUs based on radar data has not been done to the same extent as the vision based research, but there are still plenty of research on the topic. In [32] pedestrian recognition without machine learning has been studied and written about. It is shown that over 95 % of pedestrians can under optimal conditions be classified correctly with a 77 GHz radar, by primarily analysing the variance of the radial velocity of the object being classified. However, under worse conditions the classification rate can drop down to 29.4%. Laterally moving pedestrians is the main contributing factor to this drop in accuracy.

The most common approach when using deep learning methods for radar based classification is to visualise the radar data in either a Range-Doppler map or a Micro-Doppler map and then feeding this image to a CNN. This is partly done in [33] where a 25 GHz FMCW Single-Input Single-Output, SISO, radar is used in real time for human-robot identification. The CNN approach with range-Doppler maps as input is compared to conventional classical learning approaches with extracted features. In [33] only single frame range-Doppler maps are used and hence no aggregation is done.

The difference between lateral moving vehicles and pedestrians in terms of feature extraction and classification is studied in [34]. In [35] the characteristic micro-Doppler signature of pedestrians are studied with a state of the art radar sensor. Pedestrian micro-Doppler signatures are also studied in [36] together with micro-Doppler signatures of bicyclists. In [37] the micro-Doppler signatures are used as inputs to a CNN in order to classify seven different human activities. This was done with a success rate of 90.9 %. In [21] the task of semantic segmentation and classification on radar point clouds is demonstrated. The authors of [38] implements a neural network based on the MLP architecture to classify pedestrians and vehicles. This network is trained using radar outputs as input to the network.

The authors of [39] has analysed the effect of time aggregation on estimates of the elasticities of output with respect to employment and to average hours of work. They find that low frequency generate better estimates of output- employment elasticity while high frequency data generate better predicts the output-average hours elasticity. Which is a clear indicator that lower frequency not always generates better estimates or predictions, and that the hypothesis of an increasing accuracy with higher numbers of time aggregated data points might be wrong. In [40] proves both theoretically and experimentally that their proposed algorithm for the retrieval of temporal aggregates of data from sensors in infrastructures can be used to save time cost and storage space consumption. The findings in [41] show that the application of aggregation algorithms, which generalise the weighted majority algorithm, performs very well in comparison to the auto-regressive moving average algorithm. Time aggregation is mainly used in fields of economics and is not as commonly applied in the field of radar and VRU detection and classification.

3

Method

In this section the methods implemented in this thesis are explained. This includes the structure of the different datasets and how they are fed into the networks. How data is obtained and preprocessed is explained in detail, as well as the classification problem definition itself. The section ends in an explanation of how the networks are defined and trained, and which metrics are used to evaluate them.

3.1 Datasets and their characteristics

Two different types of datasets have been created, $T_{1,s}$ and $T_{2,s}$. Dataset $T_{2,s}$ contains driving scenarios inspired by driving scenarios defined by Euro NCAP conditions for a five-star rating year 2020 [1]. The driving scenarios for $T_{2,s}$ are defined in A.3 and dataset $T_{1,s}$ contains the driving scenarios defined in A.2. Hence, what distinguishes the dataset is the data points they contain. A five-fold cross-validation was done on both datasets $T_{1,s}$ and $T_{2,s}$ separately.

A study on how time aggregating data points was done on dataset $T_{1,s}$. Thus, $T_{1,s}$ was made in five different versions. One for each segment length, s, that has been tested where the variable s defines how many time frames are aggregated. The versions contain the exact same data points but the data points are aggregated over different time periods and thus have different segment lengths s.A test on the networks ability to generalise has been made. It was done by training on dataset $T_{1,s}$ and testing on dataset $T_{2,s}$.

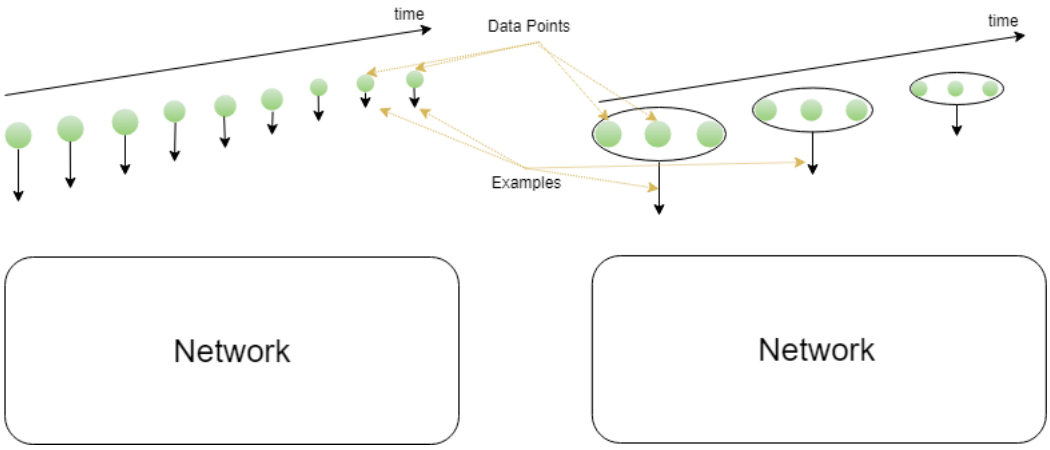


Figure 3.1: An illustration over time aggregation of data points before being fed to a network. In this figure the the segment length $s = 3$ is used as an example.

The time aggregation of data points was done with different methods for the two networks. Time aggregation of data points for the CNN was implemented by making time integrated range-Doppler maps, as will be further explained in section 3.3, and concatenation was used for the point clouds. The integration time has been set to $0.05, 0.15, 0.25, 0.35$, and 0.5 seconds. Which corresponds to the time aggregation of $1, 3, 5, 7$, and 10 data points. The time aggregation concept is illustrated in an example with segment length $s = 3$ in Figure 3.1. This means that all datasets $T_{1,s}$ contain the exact same data points when s varies, but different number of examples. This is also true for $T_{2,s}$. Note that $T_{1,s}$ and $T_{2,s}$ does not have any data points in common. As explained in Section 2.3.1, each example in a dataset is denoted $\mathbf{x}_i$. In this study $\mathbf{x}_i$ is a time integrated range-Doppler map or a point cloud, depending on the network being evaluated. The number of labels, q, for dataset $T_{1,s}$ is $q = 3$, and for $T_{2,s}$, $q = 2$. The labels are *bicyclist*, *car*, and *pedestrian* for $T_{1,s}$, and *bicyclist*, and *pedestrian* for T_2.

There are 108 889 data points in dataset $T_{1,s}$ and 42 758 data points in dataset $T_{2,s}$. The number of examples n in the datasets for varying segment length is given by Table 3.1. $T_{2,s}$ was only made with the segment lengths that yields the best results for each network.

Table 3.1: The number of examples in each dataset $T_{1,s}$ and $T_{2,s}$ with each segment length s. Dataset $T_{2,s}$ is only made in with two different segment lengths since it is only tested for with the best performing segment lengths on $T_{1,s}$.

s	1	3	5	7	10
$n_{1,s}$	108 889	37 932	22 534	16 650	11 022
$n_{2,s}$	-	14 297	-	-	4 102

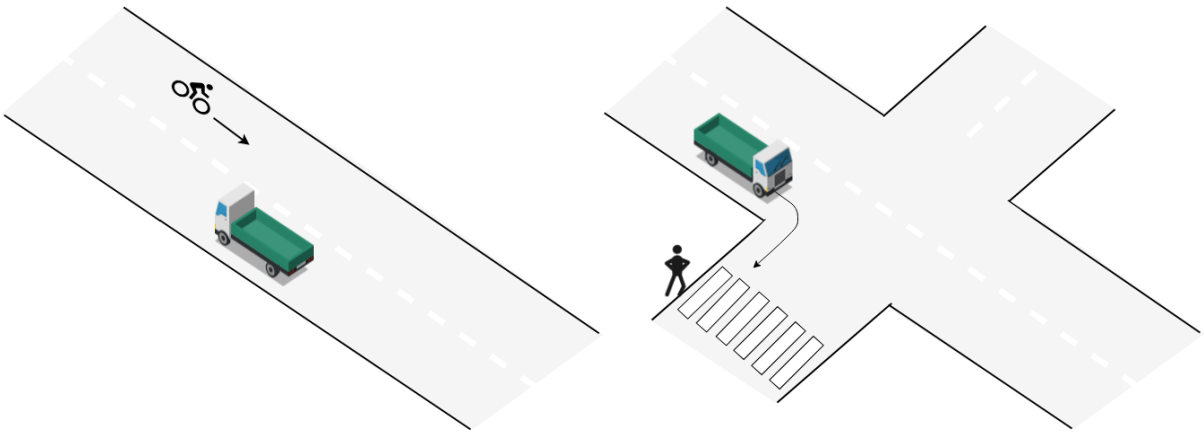
The distribution of classes in the two datasets is given by Table 3.2.

Table 3.2: The table contains the percentage of each class in the datasets $T_{1,s}$ and $T_{2,s}$.

	bicyclist	*car*	*empty*	*pedestrian*
$T_{1,s}$	21.8 %	19.1 %	32.7 %	26.4 %
$T_{2,s}$	38.5 %	0 %	33.3 %	28.2 %

3.2 Retrieval and preparation of data

The two datasets, $T_{1,s}$ and $T_{2,s}$, consists of data collected from 16 different driving scenarios. $T_{1,s}$ includes 10 of these scenarios and $T_{2,s}$ the remaining 6 scenarios. The driving scenarios in $T_{1,s}$ partly consists of data collected where the host vehicle is stationary and partly where the host is moving. In all 16 scenarios there is only one target present. This means that the scenarios are relatively simple and hence can at most be considered to be simulations of traffic scenarios with very low amount of surround targets. For every driving scenario multiple logging sessions are made. These logging sessions contain variations in distance and relative velocity between the host vehicle and the target object. Figure 3.2 illustrates one example driving scenario from each dataset. The full details of these driving scenarios can be further studied in Appendix A.2.



(a) Driving scenario 1, bicyclist. **(b)** Driving scenario 12, pedestrian

Figure 3.2: Two driving scenarios. Figure 3.2a illustrates a driving scenario from $T_{1,s}$ and Figure 3.2b illustrates a driving scenario from $T_{2,s}$.

$T_{2,s}$ consists of data gathered from 6 different driving scenarios inspired by the Euro

NCAP tests for VRU detection as stated in section 3.1. Therefore these driving scenarios are produced to be more relevant for VRU protection than the scenarios in $T_{1,s}$. All targets in $T_{2,s}$ are either of the class *bicyclist* or *pedestrian*. The details about these driving scenarios can be seen in Appendix A.2 and A.3.

All data from both datasets are collected at an empty airfield. This is to reduce the amount of radar reflections from the surroundings as much as possible. In Figure 3.3 one can see how the radars are situated on the host vehicle. There are two radars mounted on each side of the truck, making the total number of radars four. Each radar has a 150°field of view [42]. Since the target objects are not visible to the radars at all times, manual labelling of all recorded scenarios has been conducted. In order to be labelled as one of the classes, the target object is needed to be at a distance of maximum 30 m from the radar in question. When the target object exceeds the 30 m range it is no longer labelled as the specific class and these parts of the recording are pruned.

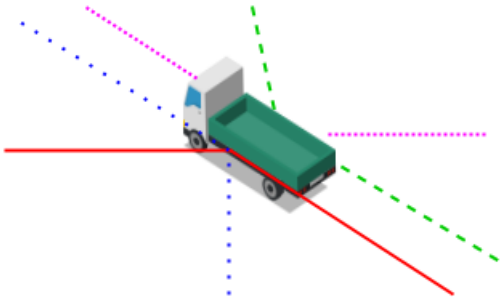


Figure 3.3: An illustration of the radar set up. The dotted lines illustrates the field of views of the radars. Each radar is represented by a specific colour.

3.3 Preprocessing of data

Each radar detection contains information of range, azimuth angle, relative velocity between the radar and the detection, and the amplitude of the received signal. This data is processed to create time integrated range-Doppler maps and point clouds.

The radar used in this thesis has a Doppler resolution of 512 and a range resolution of 128. This means that it maximally can detect 512 variations of Doppler velocities and 128 variations of range. The maximum detectable range depends on which scan type the radar is using. This particular radar has four different scan types - two mid range scans, which can detect targets up to 80 m, and two short range scans which has a maximum range of 40 m. The scan types shifts for every data point, hence the range of the radar will shift every data point. This is why a minimum range of 30 m is used when labelling the data. It is to ensure that all scan types have the target within its range.

When creating time integrated range-Doppler maps the range and Doppler resolution

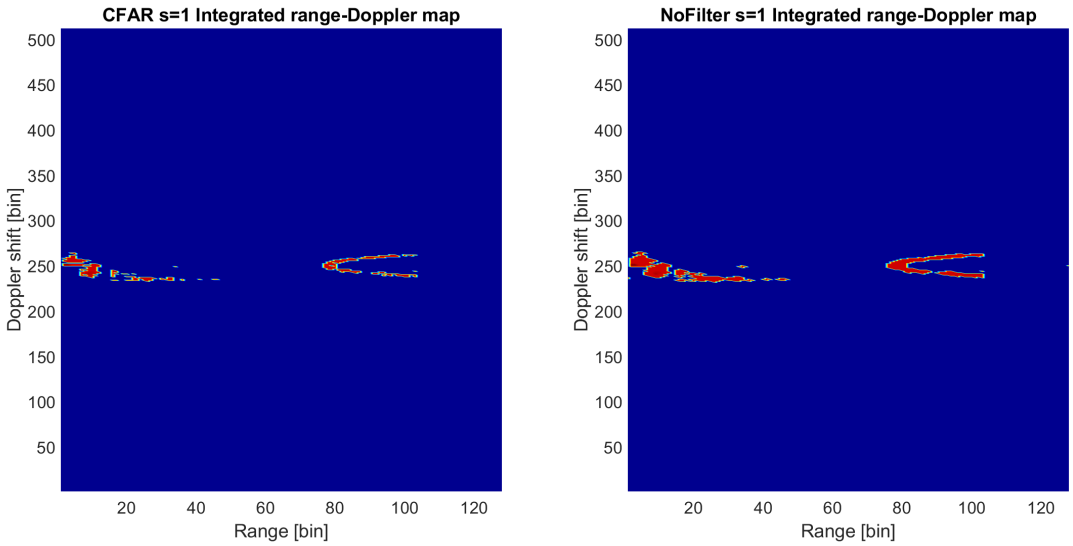
is used. The Doppler resolution is n_{D}= 512 and the range resolution is n_{r}= 128, making the images of size 512×128 pixels. A time integrated range-Doppler map is defined as I, and a logging sequence containing several range-Doppler maps is defined as M. Each detection is mapped into a pixel which has a corresponding pixel in the Range Doppler-image. The amplitude of the detected signal is used to decide the intensity of that pixel. If a detection in the next integration step maps to the same bin the intensity of the pixel is chosen to the highest value. The computations are further explained by Algorithm 2 which outputs all the time integrated range-Doppler maps in a variable K.

Algorithm 2

Integrated Range-Doppler image generation. A function returning an object with all created time integrated range-Doppler maps from one logging session. The algorithm essentially computes an addition between the range-Doppler maps being aggregated together, except if there is a value larger than 0 at the same pixel in one or more of the maps. Then it keeps the largest value. The amount of consecutive data points being aggregated together is represented by s, for segment length. Here $s = 5$ is used as an example. I is a time integrated range-Doppler map. M is a logging sequence with all range-Doppler maps from that logging, n_{D} and n_{r} are the Doppler and range resolution of the radar respectively. I is a time integrated range-Doppler map, K is the output of the function and is all the time integrated range-Doppler maps from one logging.

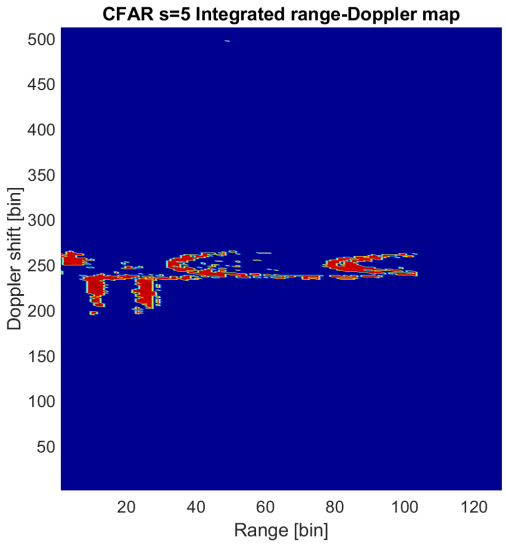
```
s = 5
M = logging with all RD-maps
n_D = 512
n_r = 128
for i = 0:/M/ do
    if (i%s) == 0 then
        I = M(i)
    end
    else
        for j = 0 : n_D do
            for k = 0 : n_r do
                if I(j,k) < M(i,j,k) then
                    I(j,k) = M(i,j,k)
                end
            end
        end
        if ((i − 1 + s)%s) == 0 then
            K.append(I)
            I = zeros
        end
    end
end
return K
```

Examples of different time integrated range-Doppler maps are shown in Figures 3.4a, 3.4b, 3.5a, 3.5b, 3.6a and 3.6b. The used segment lengths are $s = 1$, $s = 5$, and $s = 10$. The figures to the left are produced with CFAR-filtered data and the figures to the right with unfiltered data. The figures illustrates the increased information gained in the images when the segment length is increased.

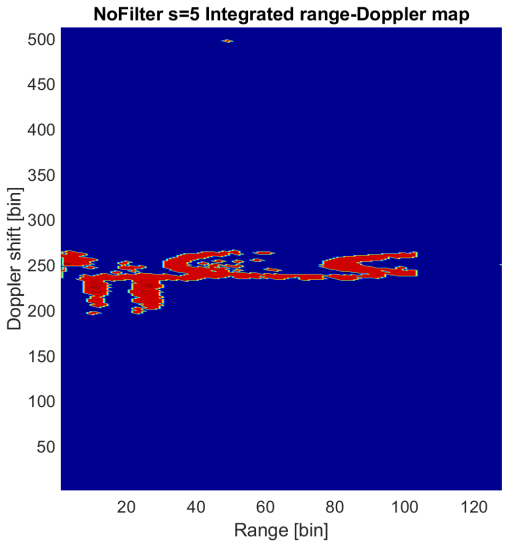


(a) Filtered range-Doppler map. **(b)** Unfiltered range-Doppler map.

Figure 3.4: Two time integrated range-Doppler maps both with segment length $s = 1$. The image to the left is created with filtered data. The filter used was a CFAR filter as explained in section 2.1.3. The image to the right is created without filtering the data.

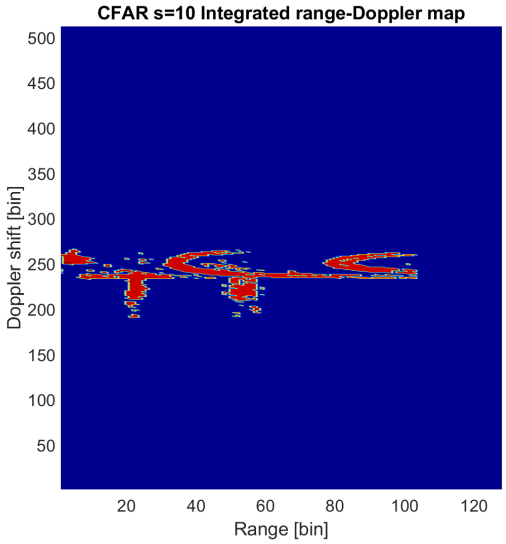


(a) Filtered range-Doppler map.

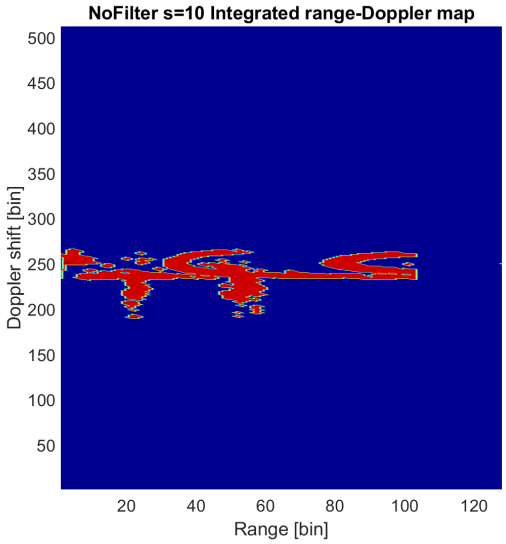


(b) Unfiltered range-Doppler map.

Figure 3.5: Two time integrated range-Doppler maps both with segment length $s = 5$. The image to the left is created with filtered data. The filter used was a CFAR filter as explained in section 2.1.3. The image to the right is created without filtering the data.



(a) Filtered range-Doppler map.

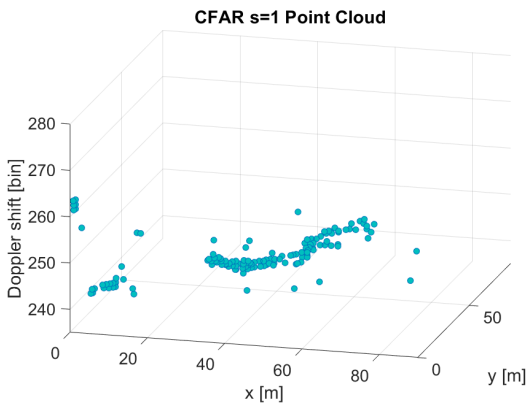


(b) Unfiltered range-Doppler map.

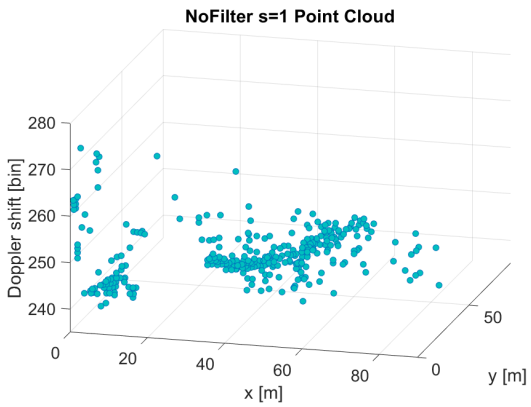
Figure 3.6: Two time integrated range-Doppler maps both with segment length $s = 10$. The image to the left is created with filtered data. The filter used was a CFAR filter as explained in section 2.1.3. The image to the right is created without filtering the data.

The point clouds were generated by concatenating x- and y-positions with the Doppler velocity, for each detection in one time frame from the radar logging. This sums up to a dimension size p= 3. The x- and y-positions are obtained by using the range and azimuth angle. The azimuth angle, θ, was computed by equation (2.3). To aggregate the point clouds over time a simple concatenation is made.

In Figures 3.7a, 3.7b, 3.8a, 3.8b, 3.9a and 3.9b each detection from one data point is plotted in 3D space. This space is defined by x- and y-axis as spacial axis in meters. The z-axis is proportional to the radial velocity of the detections. Datasets both with data filtered by the CFAR method and with no filter has been created. The figures illustrates the difference between the point clouds when filtering was used and when not. The figures to the left depicts point clouds with CFAR-filtering implemented and the figures to the right depicts point clouds where no filter is used. Figures 3.7 to 3.9 also illustrates how the data aggregation changes the form and information content of a point cloud. It is clear that curves and lines are more accentuated when the segment length s is increased.

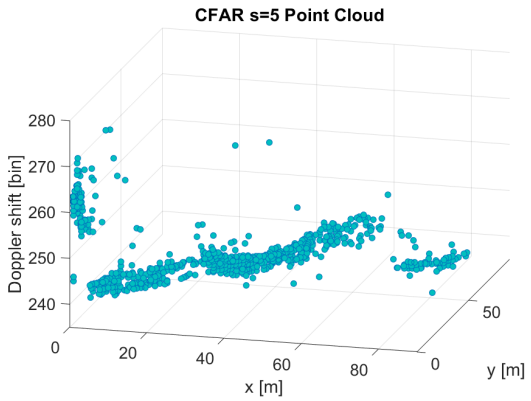


(a) Filtered point cloud.

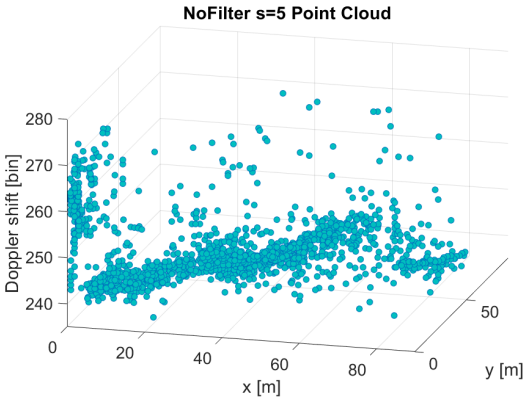


(b) Unfiltered point cloud.

Figure 3.7: A visualisation of two point clouds in 3D space where the x- and y-axis are spacial coordinates in meter and the z-axis is the Doppler shift in radar bins. The figure to the right depicts a point cloud obtained where no filtering is applied and the figure the left depicts the same point cloud but where CFAR-filtering is conducted. The point clouds are obtained with segment length $s = 1$.

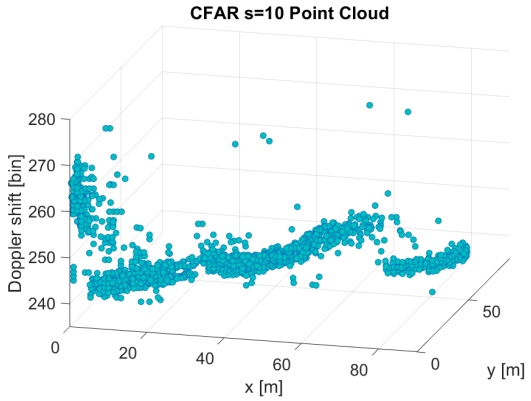


(a) Filtered point cloud.

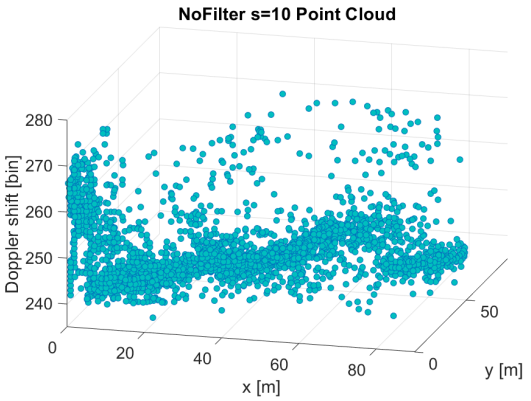


(b) Unfiltered point cloud.

Figure 3.8: A visualisation of two point clouds in 3D space where the x- and y-axis are spacial coordinates in meter and the z-axis is the Doppler shift in radar bins. The figure to the right depicts a point cloud obtained where no filtering is applied and the figure the left depicts the same point cloud but where CFAR-filtering is conducted. The point clouds are obtained with segment length $s = 5$.



(a) Filtered point cloud.



(b) Unfiltered point cloud.

Figure 3.9: A visualisation of two point clouds in 3D space where the x- and y-axis are spacial coordinates in meter and the z-axis is the Doppler shift in radar bins. The figure to the right depicts a point cloud obtained where no filtering is applied and the figure the left depicts the same point cloud but where CFAR-filtering is conducted. The point clouds are obtained with segment length $s = 10$.

Due to the full sized point clouds being too computational demanding a predefined number of points to be randomly sampled for each integration length was chosen. The sample sizes were chosen to be larger with increased segment length, but still not ending up with too large point clouds for segment length $s = 10$. In Table 3.3 the sample sizes corresponding to each integration length can be seen. Table 3.3 also contains information about how many points the average point cloud consists of for each segment length.

Table 3.3: Sample size for different integration lengths

Time aggregation length	1	3	5	7	10
Nr. sampled points	128	154	256	360	512
Mean nr. points CFAR cloud	186	557	885	1341	1911
Mean nr. points NoFilter cloud	434	1256	2238	3045	4149

3.4 Reshuffling the data

Since the data points in each dataset is a time series and therefore consecutive data points might be similar since they are close to each other in time. When creating a training and validation set it is undesirable to have neighbouring data points in time in both the validation and the training datasets, since this might yield an unfairly high accuracy. Therefore the data points have been shuffled by logging session so that data points consecutive in time never can be distributed over both the training set and validation set.

3.5 ResNet mini

The chosen CNN architecture is based on the ResNet in [3], which is a deep convolutional network based on identity mapping as explained in section 2.2.7. ResNet mini is 14 layers deep and has 67 267 parameters when the input size is 512-by-128. It is implemented Pytorch [43]. The reason why a mini-version of ResNet was chosen is partly to reduce the risk of overfitting, as described in Section 2.2.5, partly to reduce the time it takes to train the network, and partly to make it managable for the hardware to execute. The network architecture consists of a *normal block* followed by six residual blocks and a fully connected layer at the end. The architecture is illustrated in Figure 3.10.

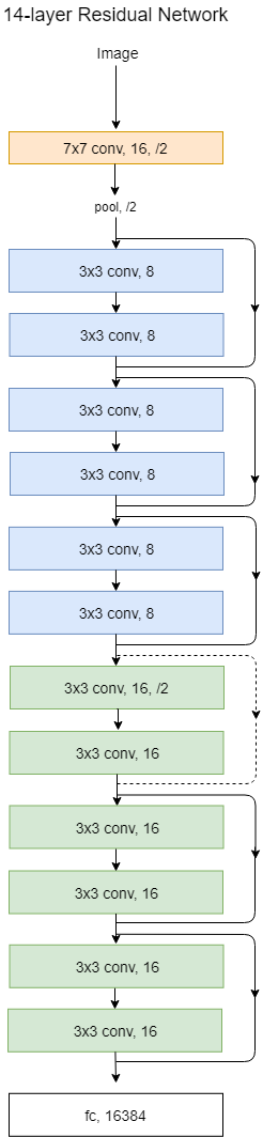


Figure 3.10: The ResNet mini architecture. The numbers denotes the size of the filters used in the layer followed by the number of filters used. In the cases where another stride than 1 is implemented it is stated at the end.

The normal block begins with a convolutional filter with kernel size 7-by-7, stride 2 and padding of 3 pixels, followed by a batch normalisation layer and a ReLu, and lastly a max pooling layer with kernel size 3-by-3, stride 2 and padding of 1 pixel.

The first three resblocks can be summarised by taking in 16 channels as input from the normal block, and outputting eight activation maps from the last of the three resblocks. In the fourth resblock the amount of filters are doubled to 16 and the input is downsampled by using a stride of 2. In order to match the dimensions for the residual mapping in this block a linear projection with a convolutional filter with kernel size 1-by-1 and stride 1 is made. This linear projection is marked by a dashed line in Figure 3.10. The second and third resblocks are identical. Both takes 16 channels as input and outputs 16 channels. The last resblock is connected to a fully

connected layer. All the layers in the resblocks consist of convolutional filters with a stride of 1, followed by a batch normalisation layer and a ReLu, except the first layer in the fourth resblock, which has a stride of 2. The parameters of the batch normalisation is defined in [3]. The code was greatly inspired by the code found in [44].

3.6 PointNet mini

The PointNet mini architecture is based on the architecture described in [4] and it was implemented in PyTorch [43]. PointNet mini is a smaller network than its predecessor. It consists of 171 386 parameters, but still keeps the main architecture of the original PointNet. PointNet mini only perform point cloud classification of complete point clouds. No segmentation is done. The reasoning behind why a mini-version is chosen for PointNet is the same as for ResNet. It is both to reduce the overfitting risk and to enhance computation times.

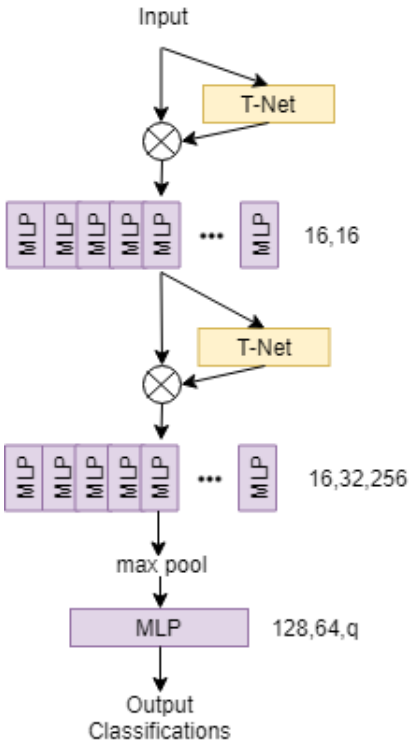


Figure 3.11: The PointNet mini architecture. The multiple stacked MLPs after each T-net module illustrates that the same MLP is used for all points. The numbers to the right represent the size of the layers in each MLP.

The architecture of PointNet mini is illustrated in Figure 3.11. The input point cloud consists of m points with dimension p. In the first *T-net* module the point cloud constructs an affine transformation matrix of size $p \times p$. In this thesis $p = 3$,

since the point clouds consists of three dimensions, as explained in Section 3.3. Each point is multiplied with the transformation matrix and used as input to the shared MLP with two layers. The layer output size is 16 for both layers, which is denoted by the numbers on the right side in Figure 3.11. The second instance of the T-net module serves the purpose of aligning the features. The network then classifies q binary classes.

3.7 Training and performance evaluation

The evaluation of both ResNet mini and PointNet mini has been done in several steps. The first step was a five-fold cross-validation of both networks on the $T_{1,s}$ dataset. A, MR and $F_{1,\mu}$ is computed for all segment lengths in order to get a good estimate of the effect that the time aggregation of data gives. Precision-Recall curves has then been done for the best performing datasets for ResNet mini and PointNet mini respectively. This gives a good indicator for how well the networks detects and classifies VRUs on $T_{1,s}$ for all segment lengths.

The reason for why Precision-Recall curves are used instead of the more commonly used ROC curve is that the ROC curve can present an overly optimistic measure of the networks performance if there are moderate to large class imbalance in the datasets, [45]. The main reason behind this is the use of False Positive Rate in the ROC curve. This is since a change in proportion between positive to negative instances does not affect the ROC curve [46]. The parameter that varies to obtain the curves is the threshold value for the binary classifiers.

In the second step of the evaluation $T_{1,s}$ was used as training set and $T_{2,s}$ as validation set. By doing this a good prediction of how well ResNet mini and PointNet mini generalise to new unseen scenarios. This evaluation was only done for the best performing segment lengths obtained when doing a five-fold cross-validation on $T_{1,s}$. In the third and final step of the evaluation a five-fold cross-validation was done on $T_{2,s}$. Precision-Recall curves have been done for both set-ups involving $T_{2,s}$ as well. The metrics in these two steps has only been computed with *pedestrian* and *bicyclist* as classes. This is to get as fair results as possible since the $T_{2,s}$ dataset does not contain any cars.

Both networks were trained for 30 epochs each and with a batch size of 32. Both networks also used Adam as optimiser. Adam was initialised with the standard settings described in 2.2.3. The classification threshold, which is the threshold value for when the binary classifier considers a probability to be true, is set to 0.5.

The networks were evaluated with data filtered with CFAR and unfiltered data. The reason for evaluating the input aspect of the problem was to ensure that important features were not lost in the filtering process. Feeding the networks with unfiltered data would eliminate one processing step and making the learning process more end-to-end compatible. In [47], benefits of end-to-end learning for self-driving cars

are described.

3.8 Computer hardware

To train a network there is preferable hardware. To run a multi-layered CNN, as the ResNet mini, a Graphics Processing Unit, GPU, is preferable. For this thesis, two different types of computers have been used. One stationary with a `GeForce GTX 1060` with 6 GB GDDR5, and two `Dell e6420` laptops with specifications in [48].

4

Results

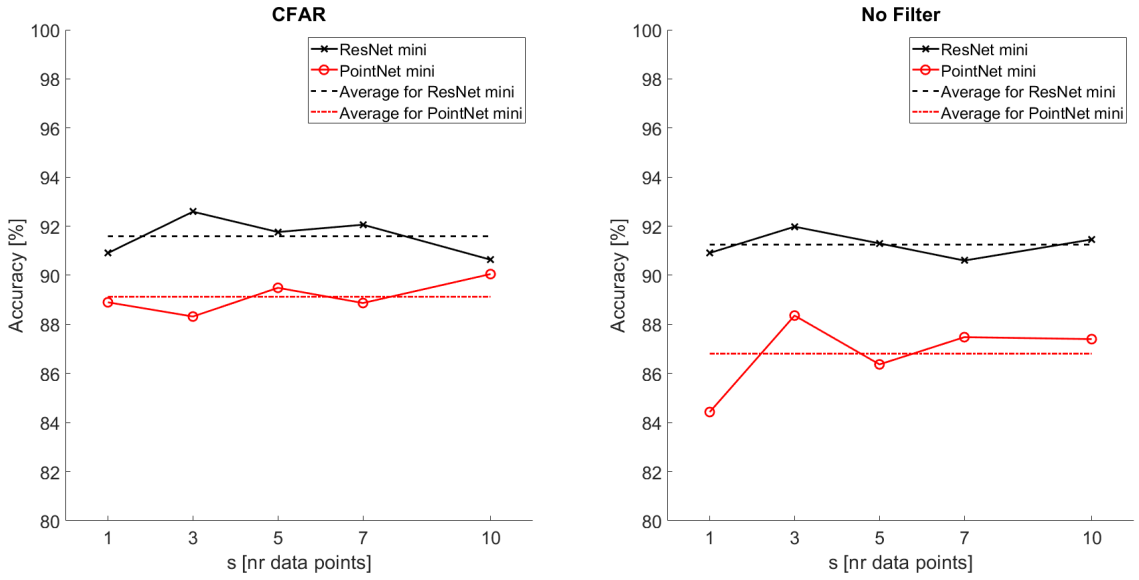
In the following chapter the results are presented. The performance of ResNet mini and PointNet mini is evaluated on two sets of data, $T_{1,s}$ and $T_{2,s}$. On the $T_{1,s}$ dataset an investigation of how the performance is effected by using time aggregated data points is done. The segment length that yields the best performance, one per network architecture, are also evaluated by five-fold cross-validation on dataset $T_{2,s}$. Lastly, the results from training on $T_{1,s}$ and using $T_{2,s}$ as validation set are presented. In summary it is shown that a five-fold cross-validation on $T_{1,s}$ and $T_{2,s}$ both yields accuracies $A > 90\,\%$. Training on $T_{1,s}$ and testing on $T_{2,s}$ on the other hand, does not yield results much better than a classifier that only makes random guesses. These results are visualised in Table 4.1.

Table 4.1: The table gives an indication of the three main results. It is viable to divide either datasets $T_{1,s}$ or $T_{2,s}$ and then train on one part of the divided dataset and test on the other. But the driving scenarios in the two datasets are too different to be able to train on $T_{1,s}$ and test on $T_{2,s}$ and get good performance.

	Results	
$T_{1,s}$	✓	
$T_{2,s}$	✓	
$T_{1,s} \rightarrow T_{2,s}$		✗

4.1 Five-fold cross-validation of $T_{1,s}$

It can be seen in Figures 4.1a and 4.1b that the general performance in accuracy is higher for ResNet mini than for PointNet mini. The mean accuracy for the filtered and unfiltered data for ResNet mini is 91.59 % and 91.25 %, and for PointNet mini it is 89.13 % and 86.81 %. This means that the ResNet mini, using filtered data for any segment length performs on average 2.46 percentage points, pp, better than the PointNet mini, and 4.44 pp for the unfiltered data.



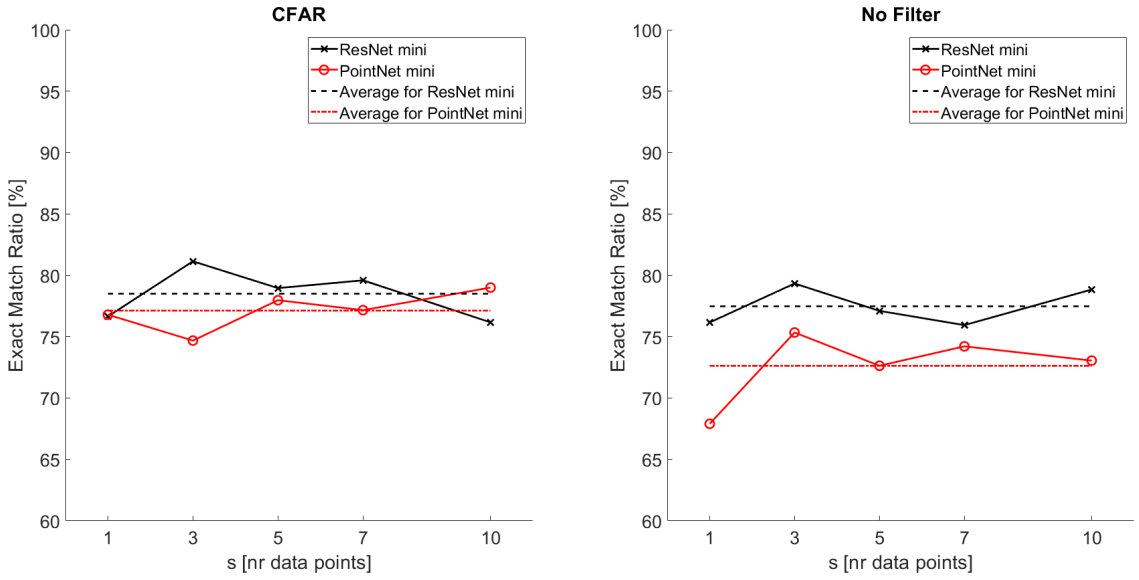
(a) **(b)**

Figure 4.1: The figures illustrates the change in accuracy A, as defined in section 2.3, over the aggregation time of the data points. The value is the average achieved value from the five-fold cross-validation. The aggregated points corresponds to a Δt of 0.05, 0.15, 0.25, 0.35, and 0.50 seconds.

Figures 4.1 to 4.3 shows that segment length $s = 3$ yields the highest scores for all metrics, for ResNet mini using both filtered and non-filtered data, and for PointNet mini when using non-filtered data. The best score overall for the PointNet mini however is obtained with segment length $s = 10$ and using filtered data. The best performing network of all is ResNet mini using three aggregated data points and filtered data. Note that the difference between using filtered and non-filtered data is marginal for ResNet mini, but substantial for PointNet mini. The scores for A, MR, and $F_{1,\mu}$ are 92.59 %, 81.14 %, and 0.83 respectively for the ResNet mini using $s = 3$ with filtered data, and 90.05 %, 79.00 %, and 0.771 for the same score for the PointNet mini using $s = 10$ with filtered data. The results for the best performing configurations are summarised in Table 4.2.

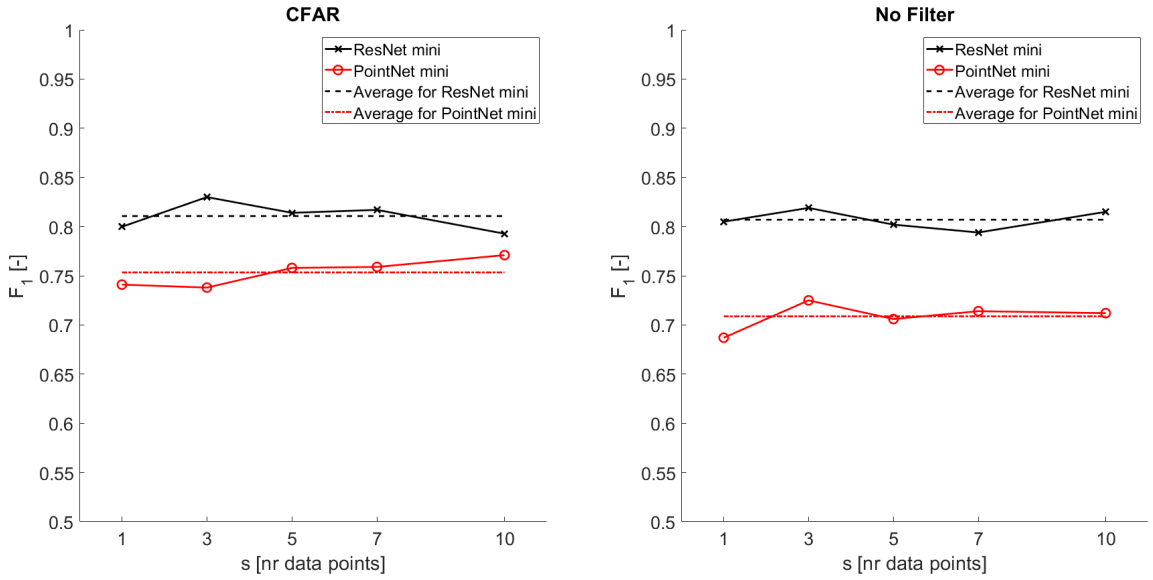
Table 4.2: The performance of the two networks with the datasets yielding the highest scores, $T_{1,3}$ and $T_{1,10}$ respectively. The results are obtained by doing a five-fold cross-validation.

	A	MR	$F_{1,\mu}$
ResNet mini	92.59	81.14	0.830
PointNet mini	90.05	79.00	0.771



(a) **(b)**

Figure 4.2: Graphs of the change in exact match ratio MR, as defined in section 2.3, over the aggregation time of the data points. The value is the average achieved value from the five-fold cross-validation. The aggregated points corresponds to a Δt of 0.05, 0.15, 0.25, 0.35, and 0.50 seconds.



(a) **(b)**

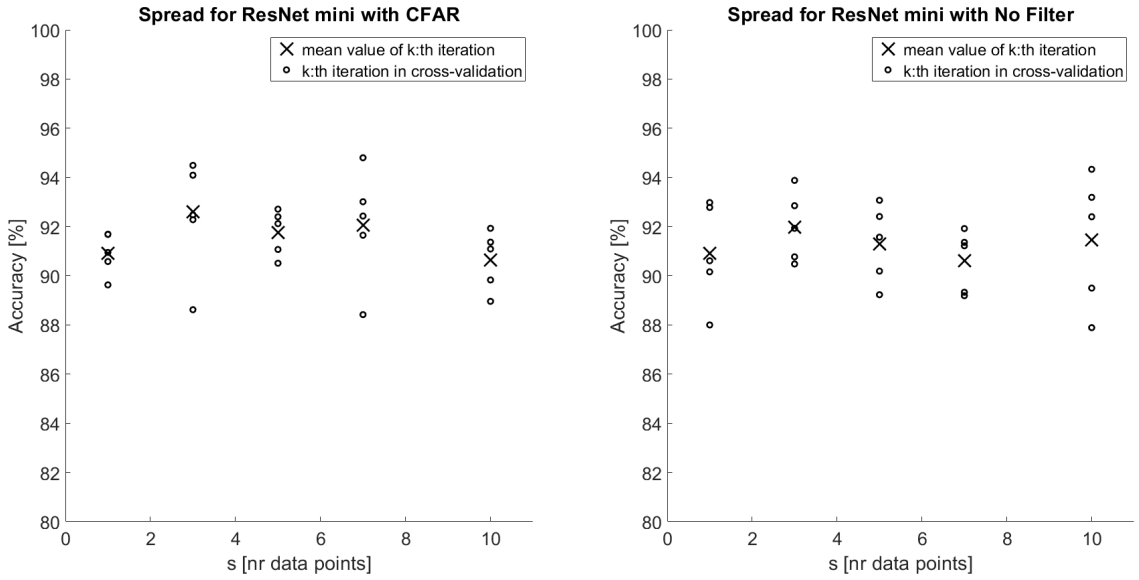
Figure 4.3: The change in $F_{1,\mu}$-score, as defined in section 2.3, over the aggregation time of the data points. The value is the average achieved value from the five-fold cross-validation. The aggregated points corresponds to a Δt of 0.05, 0.15, 0.25, 0.35, and 0.50 seconds.

When doing a five-fold cross-validation a certain spread of the different metrics is obtained. The spread of the accuracy is shown in Figures 4.4 and 4.5. The spreads are defined by the maximum and minimum value of the accuracies obtained from a certain segment length and are shown in Table 4.3 for both the ResNet mini and PointNet mini, with and without filtering of data.

Table 4.3: The maximum and minimum spreads of accuracies when doing a five-fold cross-validation. The spread is given in pp.

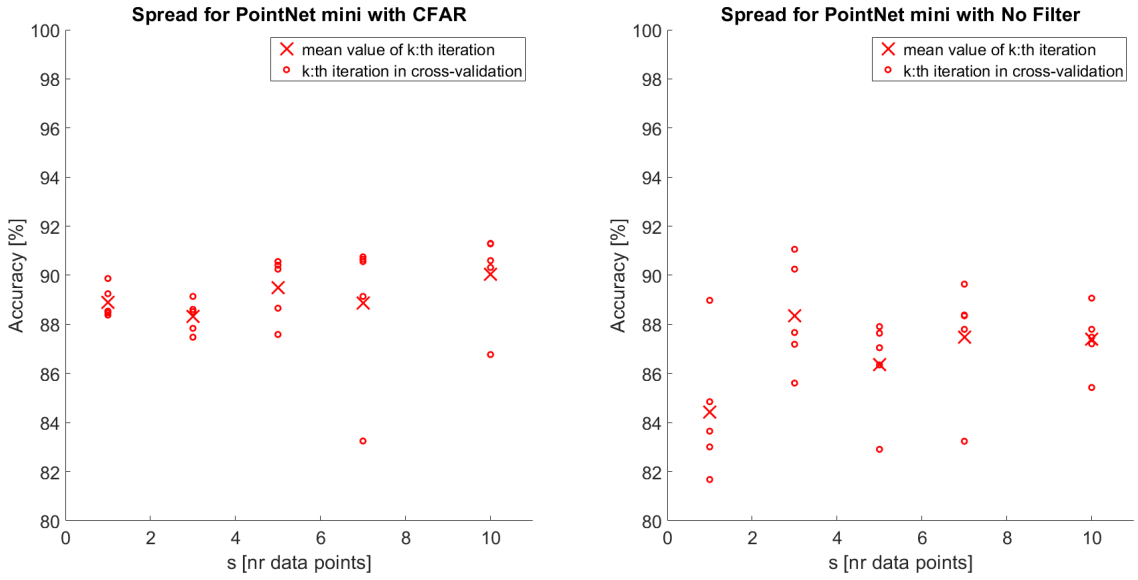
	ResNet mini		PointNet mini	
	max	min	max	min
CFAR	6.38	2.06	7.50	1.49
No Filter	6.44	2.73	7.30	3.64

In Table 4.3 the maximum and minimum values for the ResNet mini with filtered data is obtained using $s = 7$ and $s = 1$ respectively. Without filtering the data it is obtained with $s = 10$ and $s = 7$. The corresponding results for the PointNet mini, in the same order, are obtained with $s = 7, 1, 1$ and 10.



(a) **(b)**

Figure 4.4: All accuracies obtained when doing a five-fold cross-validation on the ResNet mini for segment lengths $1, 3, 5, 7$, and 10. The image to the left shows the accuracies obtained when feeding the network CFAR-filtered data and the image to right when feeding the network unfiltered data.



(a) **(b)**

Figure 4.5: All accuracies obtained when doing a five-fold cross-validation on the PointNet mini for segment lengths $1, 3, 5, 7$, and 10. The image to the left shows the accuracies obtained when feeding the network CFAR-filtered data and the image to right when feeding the network unfiltered data.

4.1.1 Precision-recall curves and *AUC*-scores

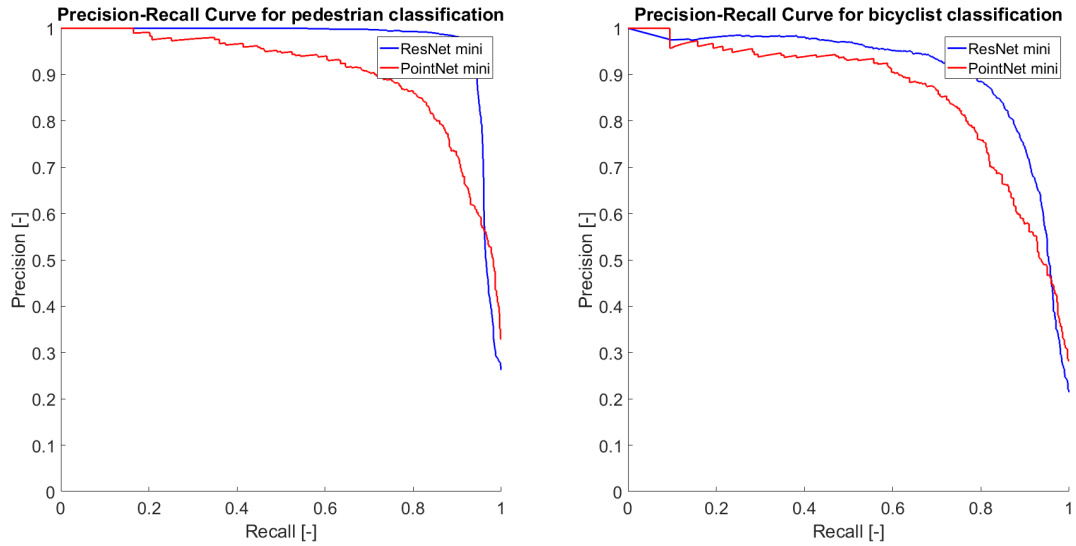


Figure 4.6: Precision-recall curves for the two evaluated networks, with a dataset using segment length $s = 3$ for ResNet mini and segment length $s = 10$ for PointNet mini, for the two classes of VRUs, *pedestrian* and *bicyclist*.

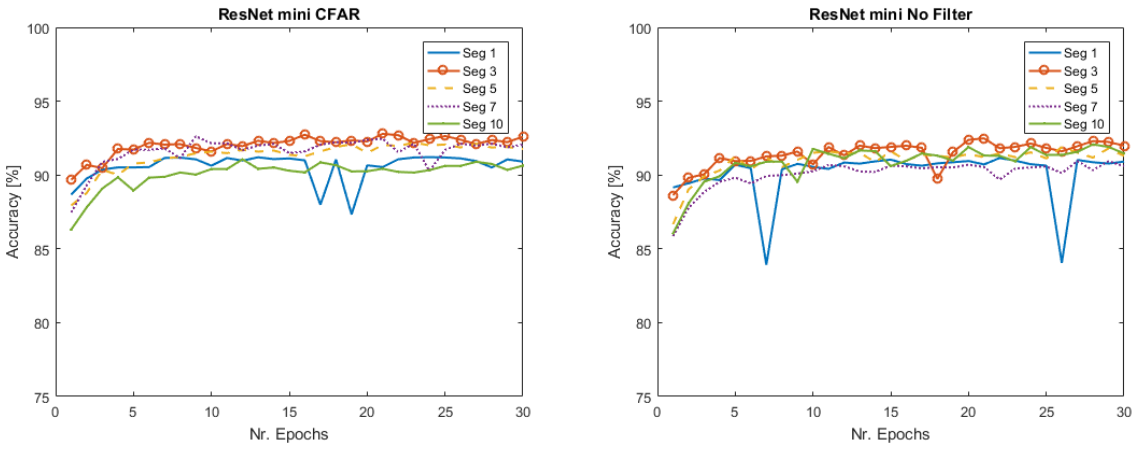
Figure 4.6 shows the precision-recall curves of ResNet mini and PointNet mini for the segment length yielding the best performance. It is shown that ResNet mini performs better than PointNet mini for almost all thresholds for both *pedestrian* and *bicyclist*. PointNet mini is only slightly better for very low threshold values for both classes. The *AUC*-scores are presented in Table 4.4, where it is clear that the ResNet mini is performing better than the PointNet mini for the class *pedestrian* and slightly better for the class *bicyclist*.

Table 4.4: *AUC*-scores for the ResNet mini and PointNet mini for the classes *pedestrian* and *bicyclist*.

	Pedestrian	Bicyclist
ResNet mini	0.968	0.905
PointNet mini	0.903	0.856

4.1.2 Training convergence rate of $T_{1,s}$

Figure 4.7a and 4.7b shows an illustration of the rate of which ResNet minis performance converge. The accuracy values of each dataset is the average value for all five fold iterations. Both figures show that ResNet mini converge after approximately 5 epochs. After 5 epochs there is no substantial incline in accuracy for any of the datasets.

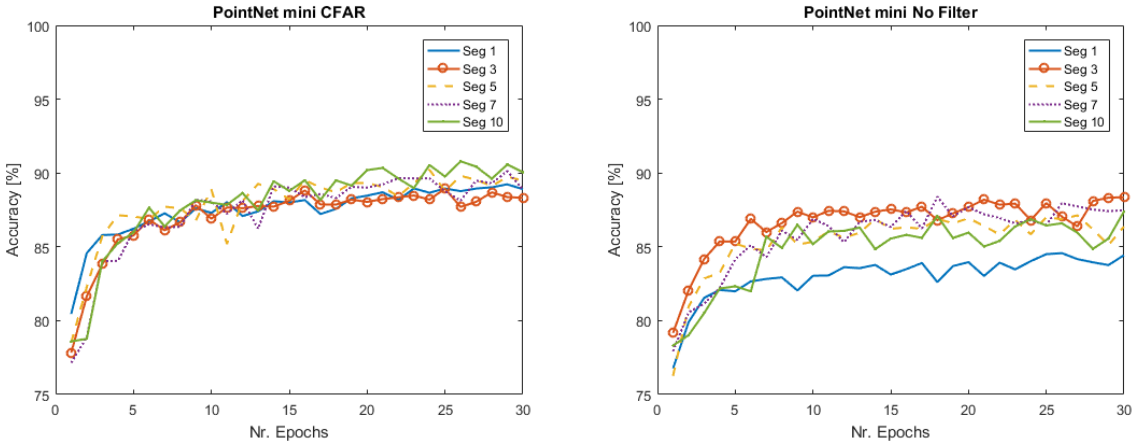


(a) ResNet mini with CFAR-filtered data **(b)** ResNet mini with not filtered data

Figure 4.7: The figures illustrates the training convergence for ResNet mini on the CFAR-filtered and unfiltered datasets. The change in accuracy, A, is shown over trained epochs for the datasets consisting of 1, 3, 5, 7 and 10 aggregated data points.

In Figure 4.8a and 4.8b the corresponding illustrations for convergence rates are shown for PointNet mini. PointNet mini is shown to require more epochs of training than ResNet mini before stagnating in performance. At which epoch number

PointNet mini converge vary depending on which dataset the network is trained on. For example, PointNet minis performance on the CFAR dataset consisting of 10 aggregated data points still shows a slight incline in performance up to epoch 30, while the unfiltered data set of 10 aggregated datapoints does not show a similar development. For the unfiltered data there is no significant rise in performance after epoch 10.

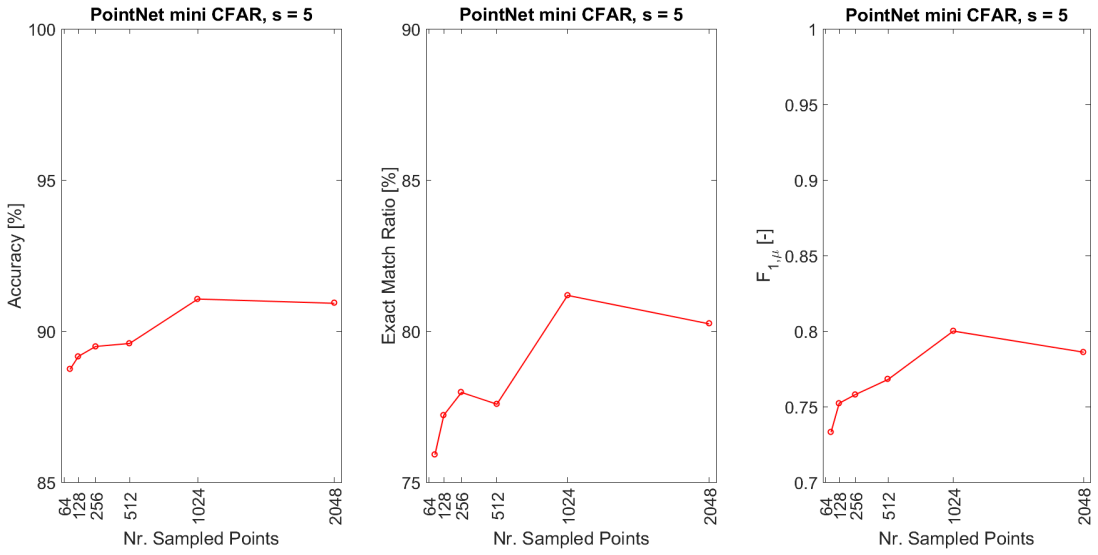


(a) PointNet mini with CFAR-filtered data **(b)** PointNet mini with not filtered data

Figure 4.8: The figures illustrates the training convergence for PointNet mini on the CFAR-filtered and unfiltered datasets. The change in accuracy, A, is shown over trained epochs for the datasets consisting 1, 3, 5, 7 and 10 aggregated data points.

4.1.3 PointNet sample size effect

Due to the restrictions set by hardware it was not possible to use the complete point clouds when training PointNet mini. Instead a sample of the point clouds was used. Figures 4.9a, 4.9b and 4.9c illustrates the effect that sample size has on the performance of PointNet mini. The metrics seen in the figures are the average values from a five fold cross validation on the CFAR dataset for five time aggregated datapoints. As described in Table 3.3 the average point cloud of this configuration consists of 885 points. All three figures shows that an increased sample size enhances the performance of the network. However, when exceeding the average point cloud size the performance seems to stagnate.



(a) **(b)** **(c)**

Figure 4.9: The figures illustrates the impact that sample size have on the performance of PointNet mini. The change in accuracy A, exact match ratio MR and $F_{1,\mu}$ is shown over number of sampled points.

4.2 $T_{1,s}$ as train set and $T_{2,s}$ as validation set

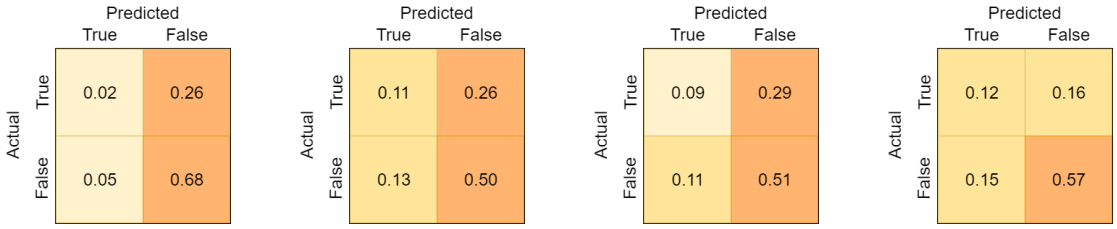
The obtained results when $T_{2,s}$ is used as validation set and $T_{1,s}$ as training set can be seen in Table 4.5 and in Figures 4.10a, 4.10b, 4.10c and 4.10d. The two networks are trained and validated on the time aggregation length resulting in best performance at the five fold cross-validation of $T_{1,s}$. Hence, PointNet mini is trained and validated on time aggregation length 10 and ResNet mini on time aggregation length 3. The results from the five fold cross-validation of $T_{1,s}$ are described and visualised in Section 4.1.

Table 4.5: The performance of the two networks using the datasets yielding the highest scores, $T_{1,3}$ and $T_{1,10}$ to train on, and validating on the corresponding datasets $T_{2,3}$ and $T_{2,10}$.

	A	MR	$F_{1,\mu}$
ResNet mini	65.04	37.64	0.264
PointNet mini	64.63	37.69	0.370

The confusion matrices in Figure 4.10 all show the same behaviour of executing a majority of false predictions. When predicting true however, all four classifiers is more likely to be wrong than correct. In total ResNet mini does 70% correct predictions for the *bicyclist*-classifier and 61% for the *pedestrian*-classifier. PointNet

mini does 60% correct classifications for *bicyclist* and 69% correct classifications for *pedestrian*.



(a) ResNet mini confusion matrix heatmap for *bicyclist*

(b) ResNet mini confusion matrix heatmap for *pedestrian*

(c) PointNet mini confusion matrix heatmap *bicyclist*

(d) PointNet mini confusion matrix heatmap for *pedestrian*

Figure 4.10: Confusion matrices for the classes *bicyclist* and *pedestrian*. The two matrices on the left are the results of ResNet mini and the two matrices on the right are the results of PointNet mini. The values in the confusion matrices correspond to the fraction of the total number of executed classifications.

The precision-recall curves in Figure 4.11 show that both PointNet mini and ResNet mini have difficulties in detecting *pedestrians* for all threshold values. PointNet mini is shown to be slightly better with a better precision for the *pedestrian* class. Both networks performs a bit better on *bicyclist* classifications, with a tiny advantage for ResNet mini. In Table 4.6 the *AUC*-scores for the respective curve is shown.

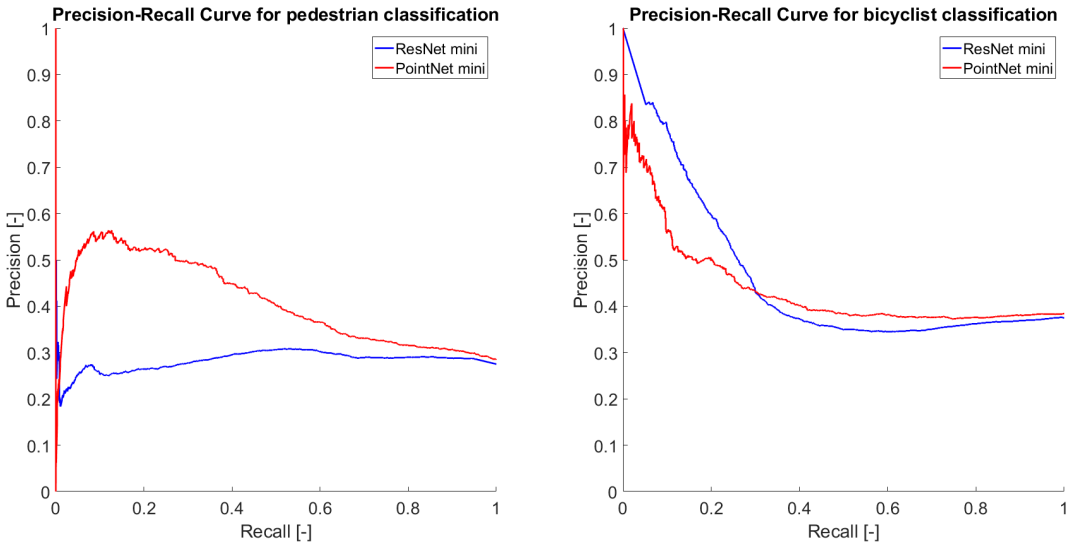


Figure 4.11: Precision-recall curves for the two classes of VRUs, *pedestrian* and *bicyclist*. The precision-recall curve is done for both evaluated networks, with a dataset using a segment length $s = 3$ for ResNet mini and a segment length $s = 10$ for PointNet mini.

Table 4.6: *AUC*-scores for ResNet mini and PointNet mini for the classes *pedestrian* and *bicyclist*. The scores are achieved after the networks have been trained on $T_{1,s}$ and validated on $T_{2,s}$.

	Pedestrian	Bicyclist
ResNet mini	0.283	0.461
PointNet mini	0.407	0.438

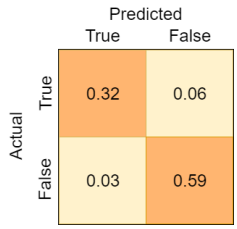
4.3 Five fold cross-validation of $T_{2,s}$

In Table 4.7 and Figures 4.12a, 4.12b, 4.12c and 4.12d the acquired results when evaluating the networks by doing a five fold cross-validation on $T_{2,s}$. The same datasets are used as in Section 4.2, hence aggregation length 10 is used for PointNet mini and aggregation length 3 for ResNet mini.

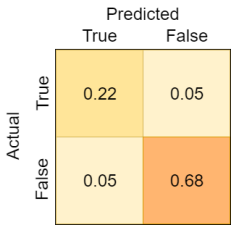
Table 4.7: The performance of the two networks with the datasets yielding the highest scores, $T_{2,3}$ and $T_{2,10}$ respectively. The results are obtained by doing a five-fold cross-validation.

	A	MR	$F_{1,\mu}$
ResNet mini	90.19	84.36	0.844
PointNet mini	84.18	74.84	0.752

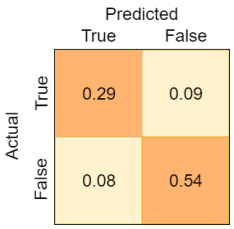
In Figure 4.12 the classification performance of the two networks are illustrated. ResNet mini correctly classifies the scenario for *bicyclist* in 91% of the cases and 90% of the cases for *pedestrian*. PointNet correctly classifies 83% of the situations for *bicyclist* and 85% of the situations for *pedestrian*.



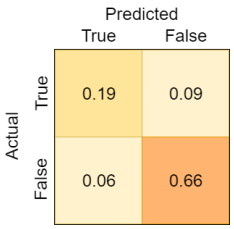
(a) ResNet mini confusion matrix heatmap for *bicyclist*



(b) ResNet mini confusion matrix heatmap for *pedestrian*



(c) PointNet mini confusion matrix heatmap *bicyclist*



(d) PointNet mini confusion matrix heatmap for *pedestrian*

Figure 4.12: Confusion matrices for the classes *bicylist* and *pedestrian*. The two matrices on the left are the results of ResNet mini and the two matrices on the right are the result of PointNet mini. The values in the confusion matrices correspond to the fraction of the total number of executed classifications during the five fold cross-validation.

In Figure 4.13 the precision-recall curves for both *pedestrian* and *bicyclist* classification are shown. ResNet mini is shown to perform better for almost all threshold values for both *pedestrian* and *bicyclist* classification. Table 4.8 contains the *AUC*-scores for the respective curve.

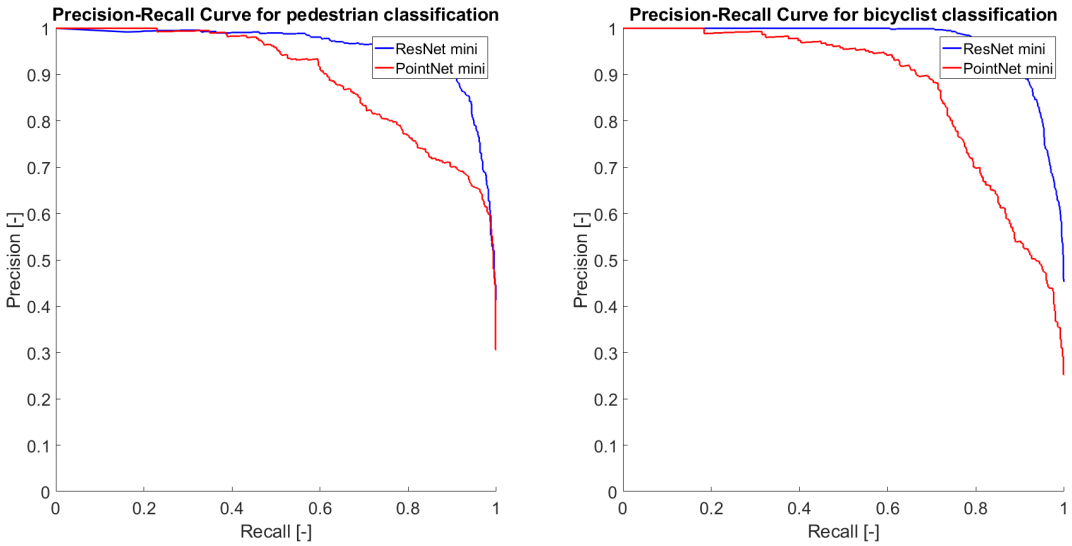


Figure 4.13: Precision-recall curves for the two classes of VRUs, *pedestrian* and *bicyclist*. The precision-recall curve is done for both evaluated networks, with a dataset using a segment length $s = 3$ for ResNet mini and a segment length $s = 10$ for PointNet mini.

Table 4.8: *AUC*-scores for ResNet mini and PointNet mini for the classes *pedestrian* and *bicyclist*. The scores are achieved after training and validation on $T_{2,s}$

	Pedestrian	Bicyclist
ResNet mini	0.956	0.969
PointNet mini	0.894	0.868

5

Discussion

The following sections begins by making a comparison between the two network architectures that have been studied. It continues on to a discussion about the effect of time aggregating data points, followed by a discussion about the datasets and how the structure of them may have affected the performance of the networks. Further the effect of filtering the data and how this might have affected the results is discussed. Lastly, potential future work is suggested.

5.1 Network comparison and performance

The results show that both networks and all segment lengths give a good result with above 84 %, 74 %, and 0.7 for the metrics A, MR, and $F_{1,\mu}$ when validated on the same type of driving scenarios as trained on. When validated on other driving scenarios however, the results are just above 64 %, 37 %, and 0.26 for A, MR, and $F_{1,\mu}$. The ResNet mini outperforms the PointNet mini when comparing the results regarding the different segment lengths. It is worth to keep in mind that a network that is guessing randomly and equally often a positive class as a negative would achieve an accuracy of $A = 50\,\%$, an exact match ratio of $MR = 12.5\,\%$, and a micro F_1-score of $F_{1,\mu} = 0.3$. These numbers are worth keeping in mind when evaluating the performance of the two networks on the different datasets.

It can be seen in Table 4.5 that the scores for the networks when training on T_1 and evaluating on T_2 are not as good as when training on the same type of driving scenarios. A slight ability to generalise to new unseen driving scenarios can however be observed for both networks. This is since the performance when only using T_2 as validation set is better than a completely random predictor. This is confirmed by all used metrics. However, the substantial increase in performance is obtained when training on driving scenarios of the same kind, as is done in Section 4.3. A plausible conclusion from this behaviour is that both networks learns the driving scenarios and not specifically learn to identify if the radar data contains a pedestrian or a bicyclist.

The precision-recall curves for both classes of VRUs shows that ResNet mini performs better than PointNet mini when training on the same driving scenarios as is

used for validation. This can be seen in Figures 4.6 and 4.13 and is emphasised by the respective *AUC*-score. Both figures shows that ResNet mini is able to achieve good precision while simultaneously maintaining high recall. It is hence possible to have fairly low threshold values and still having high precision in the predictions. This means that it is possible to detect a majority of the VRUs without over-classifying and predicting everything to be a VRU. For PointNet mini the trade-off between precision and recall is more notable. The advantage of ResNet mini is not as apparent when $T_{1,s}$ is used as training set and $T_{2,s}$ as validation set, which is illustrated in Figure 4.11. Both PointNet mini and ResNet mini have major difficulties in *pedestrians*-classification. For ResNet mini the precision even declines a bit for increased threshold values.

Regarding the training convergence for both networks, which is illustrated in Figures 4.7a, 4.7b, 4.8a and 4.8b, and explained in section 4.1, it concludes that 30 epochs of training is enough. Both networks have a rapid initial learning curve and are able to make decent prediction already after a few epochs of training. It is possible to argue that PointNet mini should be trained for more than 30 epochs to achieve the best performance possible, since it has not stagnated completely for several of the datasets. The most substantial gain in performance is however achieved during the first 10 epochs for both ResNet mini and PointNet mini. There is therefore no reason to suspect that the comparison between the two networks with 30 epochs of training was unjust.

5.2 Effect of time aggregation

The key hypothesis regarding the time aggregation of multiple data points was that an example potentially would contain a new feature that is created over time which a single data point could not contain and display. This would then make it easier for a network to make a prediction. Observing the scores in Section 4.1 this has shown to not be obviously true. No conclusion of benefits due to time aggregation of data points can be drawn. To do this a 50-fold cross-validation could be done instead. 50 values for each segment length would plausibly be more useful to make a statistical conclusion about the change in performance when the segment length is changed. Unfortunately 50 training sessions could not be done since the training sessions would be around ten times more time consuming.

ResNet mini peaks at $s = 3$ and PointNet mini peaks at $s = 10$ using CFAR. The peaks however, can be considered to be marginal and small enough to not establish with certainty that these segment lengths can be considered to yield a better performance. To be able to conclude this a spread considerably smaller than the peak in the five-fold cross-validation is desirable.

The difference in the method used for aggregating the data points could possibly have caused a somewhat unfair comparison. This is since the time integrated range-Doppler maps have the potential to ignore values in some data points due to the

nature of the definition of the time integrated range-Doppler maps [9], which can be seen in Algorithm 2. For some segment lengths and some data points it might be the case that that a point is reached where more critical information is overwritten than is gained from the concatenation. The point clouds on the other hand can contain several points with the same information, compared to the images which only has so many pixel values to save information on.

Regarding the segment length chosen for this study, it could be argued that a pedestrian does not have time to display its characteristics long enough for it to be logged properly during this time. Segment length $s = 10$ corresponds to $\Delta t = 0.5\,\mathrm{s}$ and could be considered to be too short of a time. A normal walking pace of roughly 100 steps per minute would almost yield a full step in 0.5 s. The reason for why no longer segment lengths were used is that a significantly large amount of the dataset did not contain loggings where the target was within the radars field of view for longer than 0.5 s. This would almost often be the case when the target was a car with a moderate to high speed. It is worth to note that the information content of one data point and ten time aggregated data points is not necessarily increased ten fold.

5.3 Dataset

The size of the networks was chosen to fit all datasets as well as possible. The original networks are several times larger than the versions used in this thesis. This was done to prevent overfitting [15] and to make the computational time feasible with the hardware at hand, explained in section 3.8. Since the solution in number of parameters was a one-size-fits-all type of solution there might be some generalisability to be gained from decreasing the size of the networks even more. The best results for the ResNet mini was obtained with a dataset of size 37 932 data points, which gives approximately two parameters in the network per data point. It is less then ideal, and does not follow the one in ten rule, or an *EPV* of ten as suggested in [23] and [24] for regression models. Although in [49] it is concluded that the current evidence supporting the one in ten rule for binary logistic regression is weak, and severe negative effects occur in the range where $EPV \in [2, 4]$. The networks sizes were chosen under the presumption that the datasets would be bigger and thus as a balance between keeping as much as possible of the original structure of the network and keeping the amount of parameters small. The sizes of the networks should have been further decreased in order to stay in line with the literature, even though the authors of [50] conclude that the actual degrees of freedom of a neural network are less than the parameter count in a simple XOR network. The amount of parameters are estimated with an efficient Monte-Carlo method. For some tested datasets the number of parameters is several orders of magnitude larger than the degrees of freedom. The findings in [50] suggests that it is plausible that the degrees of freedom in the networks that has been used in this study are less than the amount of parameters, and therefore the large number of parameters might not be as big of a

problem as suggested in [23] and [24] and maybe not even as severe as a $EPV = 0.5$ would be as suggested in [49].

When evaluating datasets it is almost always the case that the bigger the dataset the better. A larger dataset with an increased variety of driving scenarios could increase the performance of the two networks [51]. The $T_{1,s}$ dataset is however fairly large. With no time aggregation of data points the dataset consists of more than 100 000 examples. Another disadvantage of $T_{1,s}$ is the class imbalance and the lack of variety of the examples.

Both networks train three binary classifiers where several examples in the dataset is empty and not consisting of any class at all. This means that each classifier will train on a dataset which is consisting of a substantial majority of negative classifications. This could potentially tilt the networks towards favouring a classification of a negative class. However, both networks have *AUC*-scores above 0.8 for both VRU classes when training and validating on the same type of driving scenarios.

The quality of the datasets is worth taking into account. The labelling process is tedious and it is quite easy to make mistakes when labelling. This makes it very probable that the datasets contain a significant amount of errors. The visualisation software used for labelling shows if the detection from an object is certain or less certain. It was clear that for many frames, there were no or very few certain detections. The labelling process would be way too time-consuming and hard if this was taken into account by pruning these data points. The radar data usually comes with recorded video from the logging occasions. The logging occasion that was used for this thesis did not come with recorded video, but video from other logging occasions could be used to get an idea of the scenario in which the loggings were done. Cars and motorbikes has been seen parked next to the place where the logging occurred. Sometimes well within range of the radars. Unfortunately this leads to a degradation in the quality of the datasets.

5.4 Filtering the data

As seen in Figures 4.1 to 4.3 the ResNet mini generally handles unfiltered data better than the PointNet mini. This is when considering all metrics, i.e. accuracy A, exact match ratio MR and $F_{1,\mu}$. It can also be observed and concluded from Figures 4.4 and 4.5, where it is shown that the spread is not obviously affected for the ResNet mini when changing between filtered and unfiltered data, while it seems to be the case that the PointNet mini is affected positively by the filtering. It can be observed in the performance as well as in the spread. ResNet mini displays only a slight decrease in performance when CFAR is not used, while the performance decrease is far more severe for PointNet mini. One explanation for this difference could be found in how the point clouds are processed. It is only a small subsample of the whole point cloud that is fed to the network. When not filtering the data this subsample of the point cloud is more likely to consist of pure noise.

The noise is probably random and due to noise from the Johnson effect [52] and cosmic background radiation [53], which is common noise for electronic devices and radars. CNN architectures have been found in [54] to be more susceptible to noise than to other types of distortions. The standard deviation of the noise in Figures 3.4b, 3.5b and 3.6b is 14.24, 24.91 and 24.74. Which for the networks tried out in [54] yields an accuracy loss of around 10 pp. The noise does not seem to have the same effect on ResNet mini. This might be due to the CFAR method for removing the noise since the CFAR method only applies an adaptive threshold and does not really remove all Gaussian noise. The authors of [4] and creators of the PointNet prove in the mentioned paper that the architecture is somewhat resistant to noise up to a certain threshold. Since the point clouds and the time integrated range-Doppler maps are very different in nature, it is hard to compare the standard deviation and its effect on the two.

5.5 Future work

An interesting take for future work would be to make the labelling process automated. This could be done since Aptiv has a tracker that keeps track of moving objects. A condition to only save the data for the dataset where the tracked object is within range could then be put in place. This would make it possible to make a far larger and more diverse dataset than the existing manually labelled dataset. It would for instance be possible to include much more complex scenarios involving multiple targets. These type of scenarios requires extensive work to label manually. A second condition could then be put in place to prune those loggings where there are not long enough segment lengths. In this way it would be possible to properly examine the effect of longer segment lengths. An automated labelling process would also yield the possibility of potentially labelling specific detections. This would be especially beneficial for PointNet mini. As showed in [21] PointNet is able to be trained to segment the complete point cloud input. However, as this requires individual labelling of each radar detection, an automated labelling process is a necessity for this to be implementable.

An investigation of additional features to add to the point clouds is an other area where further studies would be of interest. This is since it is the main advantage of using a PointNet architecture over a CNN based architecture. The CNN based network will always be limited by the amount of features possible to visualise in images, while PointNet is able to process the information as data with multiple dimensions.

A couple of suggestions to study to solve the problem of data being overwritten in the time integrated range-Doppler maps, as explained in section 5.2, are to either create multi-channeled inputs for the CNN, where the amount of channels is equal to the segment length. Another idea would be to simply perform an addition of all range-Doppler maps within the span of a segment length. This solution does not

contain as much information as the previous idea but it is less complex, and thus easier to implement.

Before the start of this project a thorough pre-study was made where several interesting preprocessing methods were found. For example the variance of the relative radial velocity components in the frequency image data, as shown in [32]. The information in these maps could be turned into images or point clouds with little effort, and thus used as input to ResNet mini and PointNet mini.

There are several other classification tasks to be explored. For example road classification as in [55], where a comparison between radar and image data for road type classification, using different machine learning methods, was conducted. Radar data was in [55] shown to not be applicable, but it speaks for the aspiration in the industry for radar based road classification.

Aptiv has developed new and better radars than the one used for this thesis, which can be found in [42]. It is easy to assume that newer radars open up the possibilities for a better performance in the classification of the networks. This would aswell be of interest to see a study on.

6

Conclusion

It can be concluded that the ResNet mini performs better than the PointNet mini on all evaluation metrics, except the AUC-score for the class *bicyclist*, and is more robust against noise. The effect of time aggregation needs to be studied further before any conclusion can be drawn about it. The study also shows that it is important to train on scenarios similar to the ones to be tested. That is, the networks have a significant limitation to generalise between very different driving scenarios. Hence, in order to be applicable as a real world application, a very diverse training set is necessary. Regarding the performance it is on average beneficial to filter the data points with CFAR. This study concludes that the best approach among the ones studied in this thesis is to use the ResNet mini with time integrated range-Doppler maps filtered by CFAR, with some segment length, as input. At best this approach achieves the scores $A = 92.59\,\%$, $MR = 81.14\,\%$, $F_{1,\mu} = 0.83$, $AUC_{pedestrian} = 0.968$, and $AUC_{bicyclist} = 0.905$. It is plausible that this approach could be used as part of a VRU detection system, and that the results are further improvable. Hence, the proof of concept can be considered to be proved.

Bibliography

[1] EuroNCAP. (2020) Test protocol - aeb vru systems. [Online]. Available: https://www.euroncap.com/en/for-engineers/protocols/vulnerable-road-user-vru-protection/

[2] J. Schyllander, "Fotgängarolyckor: statistik och analys," https://www.msb.se/RibData/Filer/pdf/27438.pdf, sep 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2016.

[5] B. Ballot, "Akustische versuche auf der niederländischen eisenbahn, nebst gelegentlichen bemerkungen zur theorie des hrn. prof. doppler," *Annalen der Physik und Chemie*, vol. 142, no. 11, pp. 321–351, 1845. [Online]. Available: https://doi.org/10.1002/andp.18451421102

[6] V. Issakov, *Microwave Circuits for 24 GHz Automotive Radar in Silicon-based Technologies*. Springer Berlin, 2014.

[7] N. Karlström and A. Öqvist, "Target identification using low level radar measurements," Master's thesis, 2018.

[8] T. Jeffrey, *Phased-array radar design: application of radar fundamentals*. SciTech Pub., 2009.

[9] D. Tahmoush and J. Silvious, "Visualizing and displaying radar micro-doppler data," in *Radar Sensor Technology XV*, K. I. Ranney and A. W. Doerry, Eds. SPIE, may 2011. [Online]. Available: https://doi.org/10.1117/12.883447

[10] Y. Luo, Y. Wong, M. Kankanhalli, and Q. Zhao, "$\mathcal{G}$-softmax: Improving intra-class compactness and inter-class separability of features," 2019.

[11] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," 2012.

[12] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," 2018.

[13] Y. B. Ian Goodfellow and A. Courville, "Deep learning," 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[15] I. V. Tetko, D. J. Livingstone, and A. I. Luik, "Neural network studies. 1. comparison of overfitting and overtraining," *Journal of Chemical Information and Modeling*, vol. 35, no. 5, pp. 826–833, sep 1995. [Online]. Available: https://doi.org/10.1021/ci00027a006

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[17] Sergey, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Mar 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[18] K. He and J. Sun, "Convolutional neural networks at constrained time cost," 2014.

[19] G. B. Goh, N. O. Hodas, and A. Vishnu, "Deep learning for computational chemistry," *Journal of Computational Chemistry*, vol. 38, no. 16, pp. 1291–1307, Mar. 2017. [Online]. Available: https://doi.org/10.1002/jcc.24764

[20] J. Schmidhuber, "Deep learning in neural networks: An overview," 2014.

[21] O. Schumann, M. Hahn, J. Dickmann, and C. Wohler, "Semantic segmentation on radar point clouds," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, jul 2018. [Online]. Available: https://doi.org/10.23919/icif.2018.8455344

[22] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, "Binary relevance for multi-label learning: an overview," *Frontiers of Computer Science*, vol. 12, pp. 1–2, 11 2017.

[23] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, "A simulation study of the number of events per variable in logistic regression analysis," *Journal of Clinical Epidemiology*, vol. 49, no. 12, pp. 1373–1379, Dec. 1996. [Online]. Available: https://doi.org/10.1016/s0895-4356(96)00236-3

[24] F. E. Harrell, K. L. Lee, R. M. Califf, D. B. Pryor, and R. A. Rosati, "Regression modelling strategies for improved prognostic prediction," *Statistics*

in Medicine, vol. 3, no. 2, pp. 143–152, Apr. 1984. [Online]. Available: https://doi.org/10.1002/sim.4780030207

[25] G. J. McLachlan, K.-A. D, and C. Ambroise, *Analyzing microarray gene expression data*. John Wilwy Sons, 2004, ch. 6.19.2, p. 214.

[26] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, Aug. 1998. [Online]. Available: https://doi.org/10.1016/s0968-090x(98)00019-9

[27] D. M. Gavrila, "Pedestrian detection from a moving vehicle," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, pp. 37–49. [Online]. Available: https://doi.org/10.1007/3-540-45053-x_3

[28] A. Shashua, Y. Gdalyahu, and G. Hayun, "Pedestrian detection for driving assistance systems: single-frame classification and system level performance," in *IEEE Intelligent Vehicles Symposium, 2004.* IEEE. [Online]. Available: https://doi.org/10.1109/ivs.2004.1336346

[29] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016.

[30] S. Liang, R. Sun, J. D. Lee, and R. Srikant, "Adding one neuron can eliminate all bad local minima," 2018.

[31] K. Kawaguchi and L. P. Kaelbling, "Elimination of all bad local minima in deep learning," 2019.

[32] A. Bartsch, F. Fitzek, and R. H. Rasshofer, "Pedestrian recognition using automotive radar sensors," *Advances in Radio Science*, vol. 10, pp. 45–55, sep 2012. [Online]. Available: https://doi.org/10.5194/ars-10-45-2012

[33] S. Abdulatif, Q. Wei, F. Aziz, B. Kleiner, and U. Schneider, "Micro-doppler based human-robot classification using ensemble and deep learning approaches," in *2018 IEEE Radar Conference (RadarConf18).* IEEE, apr 2018. [Online]. Available: https://doi.org/10.1109/radar.2018.8378705

[34] S. Heuel and H. Rohling, "Pedestrian classification in automotive radar systems," in *2012 13th International Radar Symposium.* IEEE, may 2012. [Online]. Available: https://doi.org/10.1109/irs.2012.6233285

[35] P. Held, D. Steinhauser, A. Kamann, T. Holdgrun, I. Doric, A. Koch, and T. Brandmeier, "Radar-based analysis of pedestrian micro-doppler signatures using motion capture sensors," in *2018 IEEE Intelligent Vehicles Symposium (IV).* IEEE, Jun. 2018. [Online]. Available: https://doi.org/10.1109/ivs.2018.8500656

[36] D. Belgiovane and C.-C. Chen, "Micro-doppler characteristics of pedestrians and bicycles for automotive radar sensors at 77 GHz," in *2017 11th European Conference on Antennas and Propagation (EUCAP)*. IEEE, Mar. 2017. [Online]. Available: https://doi.org/10.23919/eucap.2017.7928457

[37] Y. Kim and T. Moon, "Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, jan 2016. [Online]. Available: https://doi.org/10.1109/lgrs.2015.2491329

[38] S. Park, J. P. Hwang, E. Kim, H. Lee, and H. G. Jung, "A neural network approach to target classification for active safety system using microwave radar," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2340–2346, Mar. 2010. [Online]. Available: https://doi.org/10.1016/j.eswa.2009.07.070

[39] M. M. Estevão, "Measurement error and time aggregation: a closer look at estimates of output-labor elasticities," 1996.

[40] I. Manojlović and A. Erdeljan, "Efficient aggregation of time series data," 05 2017.

[41] W. Jamil, Y. Kalnishkan, and A. Bouchachia, "Aggregation algorithm vs. average for time series prediction," 2016.

[42] *Radar Sensor Profiles Algorithm Definition Document*, Aptiv, 8 2015.

[43] e. a. Edward Z. Yang, Adam Paszke, "Pytorch," https://github.com/pytorch, 2013.

[44] J. Liu, "Resnet family code," https://github.com/chisyliu/signal-processing.-machine-learning-and-sensor-fusion/blob/master/ResNet_Family_Code/ResNet_testing.py, 2013.

[45] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.

[46] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," pp. 10–11, 2004.

[47] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1604.html#BojarskiTDFFGJM16

[48] *DELL LATITUDE E6420*, Dell.

[49] M. van Smeden, J. A. H. de Groot, K. G. M. Moons, G. S. Collins,

D. G. Altman, M. J. C. Eijkemans, and J. B. Reitsma, "No rationale for 1 variable per 10 events criterion for binary logistic regression analysis," *BMC Medical Research Methodology*, vol. 16, no. 1, Nov. 2016. [Online]. Available: https://doi.org/10.1186/s12874-016-0267-3

[50] T. Gao and V. Jojic, "Degrees of freedom in deep neural networks," 2016.

[51] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *International Journal of Computer Vision*, vol. 119, no. 1, pp. 76–92, 2016.

[52] J. B. Johnson, "Thermal agitation of electricity in conductors," *Physical Review*, vol. 32, no. 1, pp. 97–109, Jul. 1928. [Online]. Available: https://doi.org/10.1103/physrev.32.97

[53] J. Lee and G. Cleaver, "The cosmic microwave background radiation power spectrum as a random bit generator for symmetric and asymmetric-key cryptography," *Heliyon*, vol. 3, 11 2015.

[54] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," 2016.

[55] J. Warnborg, "On classification of road types for automotive applications," Master's thesis, 2018.

[56] F. C. Bernardini, R. B. da Silva, R. M. Rodovalho, and E. B. M. Meza, "Cardinality and density measures and their influence to multi-label learning methods," *Dens*, vol. 1, p. 13.

[57] P. P. da Gama, F. C. Bernardini, and B. Zadrozny, "Rb: A new method for constructing multi-label classifiers based on random selection and bagging," *Learning and Nonlinear Models*, vol. 11, no. 1, pp. 26–47, 2013.

A

Appendix 1

A.1 Dataset cardinality and densitiy

The datasets characteristics are defined by the label cardinality,

$$C = \frac{1}{n}\sum_{i=1}^{n}|Y_i|, \tag{A.1}$$

and label density,

$$D = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i|}{|\mathcal{L}|}, \tag{A.2}$$

where $\mathcal{L}$ is the labelset. C is the number of positive classes per example and can therefor be as large as the number of classes in the most extreme case. D on the other hand has a maximum of 1 since it is the number of positive classes per class per example. In other words, proportion between the number of positive classes and total number of classifications to be made.

In Table A.1 C and D are defined for each dataset. These two metrics are related to how difficult it is to learn a multi-label classifier [56]. The authors in [56] conclude that a lower C and higher D tends to make the multi-label learning process more difficult, and the authors in [57] conclude that higher C and lower D tends to make the multi-label learning process easier.

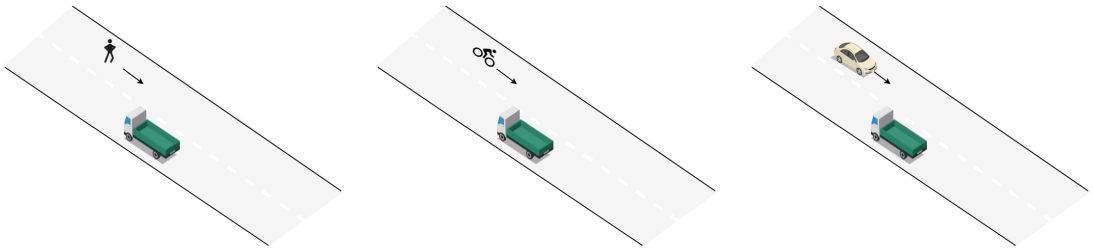
Table A.1: The cardinality C and density D of the two datasets $T_{1,s}$ and $T_{2,s}$ is presented.

	$T_{1,s}$	$T_{2,s}$
C	0.67	0.68
D	0.22	0.23

The imbalance of negative classes to positive classes in the examples in the datasets, with substantially more negative classes, probably tilts the classifiers towards classifying an ambiguous example as a negative class, since the probability of the example

actually being negative is statistically higher. This could be fixed by having several classes per example to get higher cardinality of the datasets.

A.2 Dataset T_1

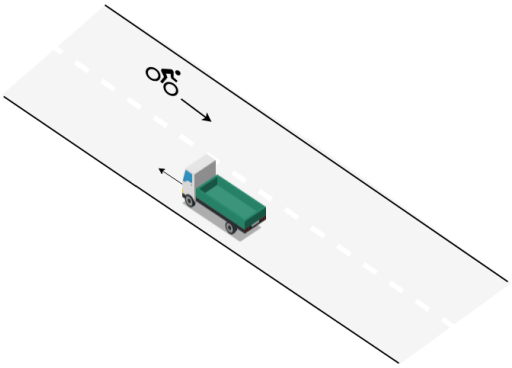


(a) Driving Scenario 1 pedestrian

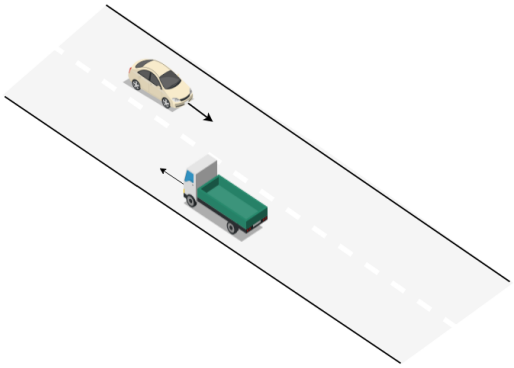
(b) Driving Scenario 1 bicyclist

(c) Driving Scenario 1 car

Figure A.1: The target object accelerates to required speed while the host vehicle remains stationary. For the scenarios involving a car the required speed varies between 20kph, 50kph, and 100kph, while the speed for the pedestrian always i set to 5kph and the speed of the bicyclist to 20kph. Multiple sessions are logged with different lateral distance to the target. The pedestrian scenarios are done with a lateral distance of 2m, 3m, 6m, 10m, 12m, 15m, 18m, and 20m. The bicyclist scenarios are done with a lateral distance of 2m, 3m, 12m, 18m, and 20m. The car scenarios are done with a lateral distance of 3m, 6m, and 10m. The scenarios are also conducted from both sides of the host vehicle and with the target travelling in both directions.

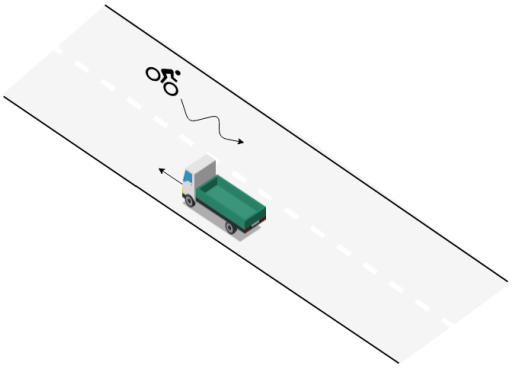


(a) Driving Scenario 2 bicyclist

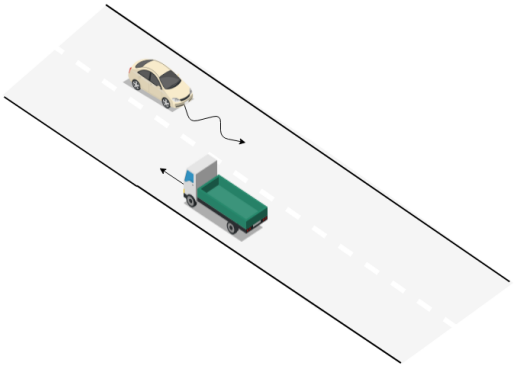


(b) Driving Scenario 2 car

Figure A.2: The target object and host vehicle accelerates to required speed. For the scenarios involving a car the speed varies between 60kph, 80kph, and 90kph while the speed for the bicyclist varies between 20kph and 30kph. The host speed is either 10kph or 50kph. The scenario contains both oncoming situations as illustrated in the figures and overtaking situations.

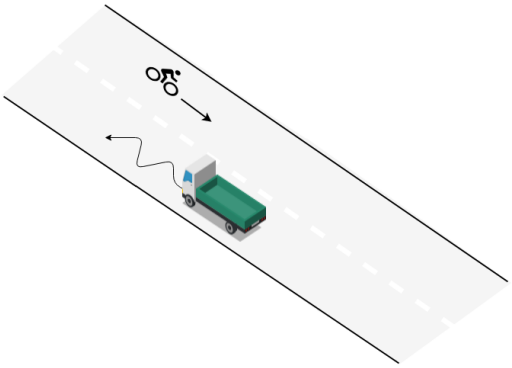


(a) Driving Scenario 3 bicyclist

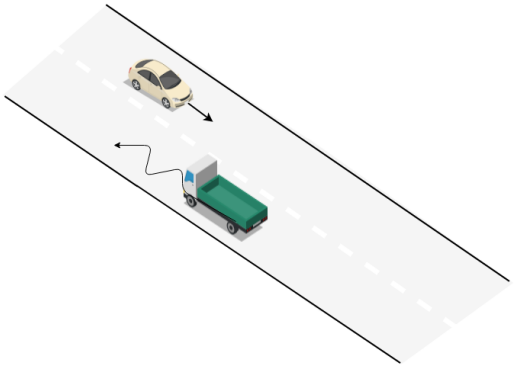


(b) Driving Scenario 3 car

Figure A.3: The target object and host vehicle accelerates to required speed. The target then drives in a slalom pattern past the host. The speed of the car varies between 30kph and 60kph, while the speed for the bicyclist is 20kph. The host speed is 10kph. The scenario contains both oncoming situations as illustrated in the figures and overtaking situations.

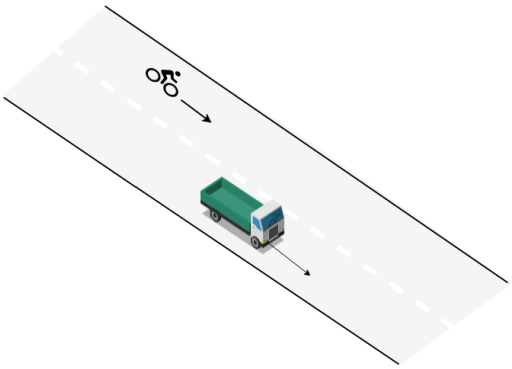


(a) Driving Scenario 4 bicyclist

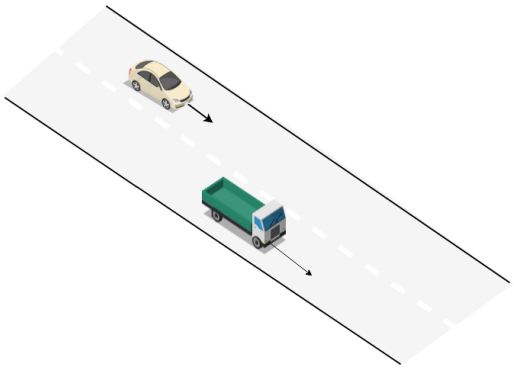


(b) Driving Scenario 4 car

Figure A.4: The target object and host vehicle accelerates to required speed. The host vehicle then drives in a slalom pattern as the target passes by. The speed of the car varies between 10kph, 50kph and 80kph, while the speed for the bicyclist is always 10kph. The host speed varies between 20kph and 50 kph for the car scenarios, but is only 20kph for the bicycle scenarios. The scenario contains both oncoming situations as illustrated in the figures and overtaking situations.

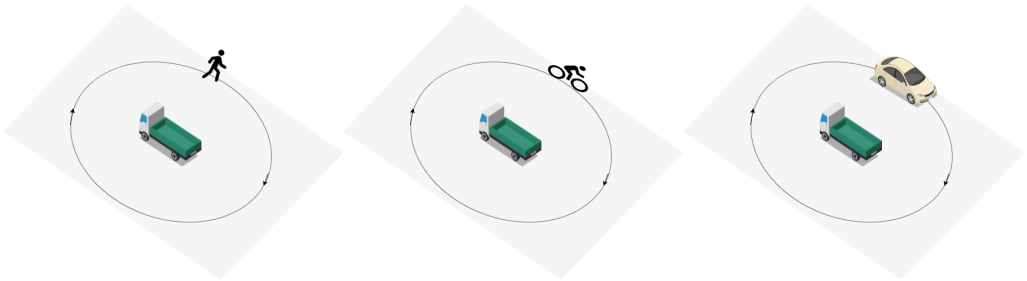


(a) Driving Scenario 5 bicyclist



(b) Driving Scenario 5 car

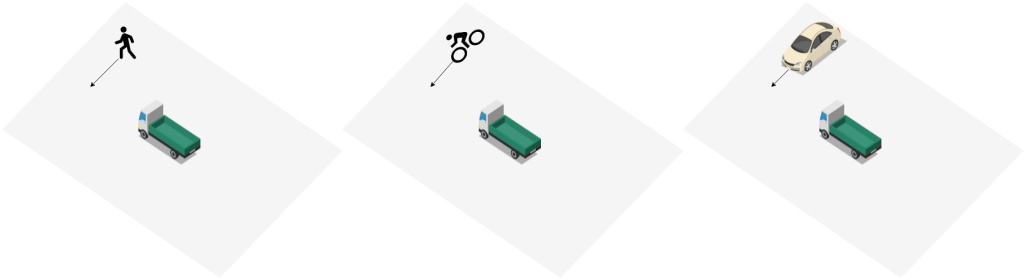
Figure A.5: The target object and host vehicle accelerates to required speed. The target then brakes to a complete stop next to the host. The speed of the car is 50kph, the speed of the bicyclist is 30kph and the speed of the host is 10kph.



(a) Driving Scenario 6 pedestrian

(b) Driving Scenario 6 bicyclist

(c) Driving Scenario 6 car

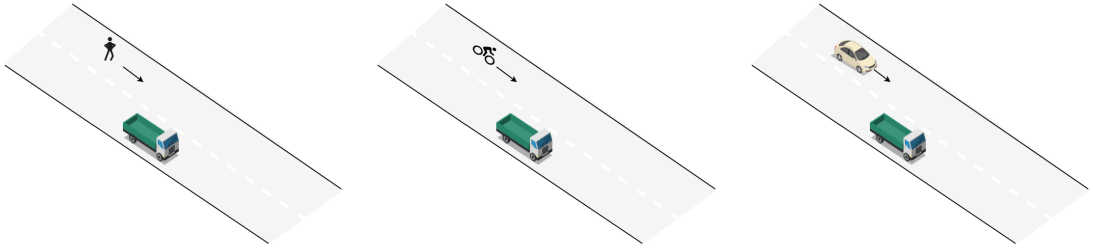


(d) Driving Scenario 6 pedestrian

(e) Driving Scenario 6 bicyclist

(f) Driving Scenario 6 car

Figure A.6: The target object accelerates to required speed while the host vehicle remains stationary. Driving scenario 10 is divided into two sub-scenarios for each target, as is illustrated. The car accelerates to 40 kph, the bicycle to 30kph and the pedestrian to 5kph. In the sub-scenarios A.6a,A.6b and A.6c the target keeps a distance of 5m throughout the logging. These sub-scenarios are done both clock-wise and counter clock-wise. The sub-scenarios in A.6d,A.6e and A.6f are done both from right to left and vice versa.

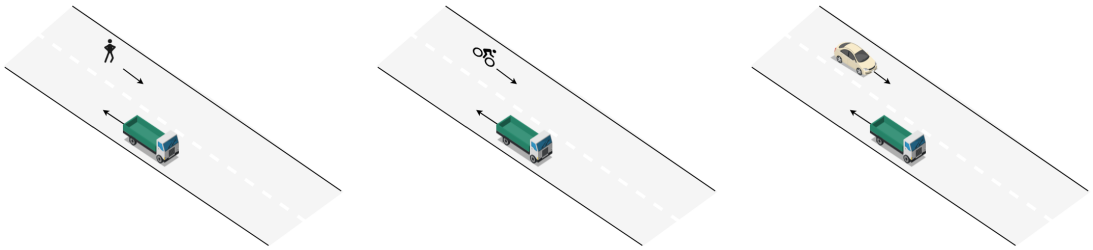


(a) Driving Scenario 7 pedestrian

(b) Driving Scenario 7 bicyclist

(c) Driving Scenario 7 car

Figure A.7: The target object accelerates to 5kph while the host vehicle remains stationary.

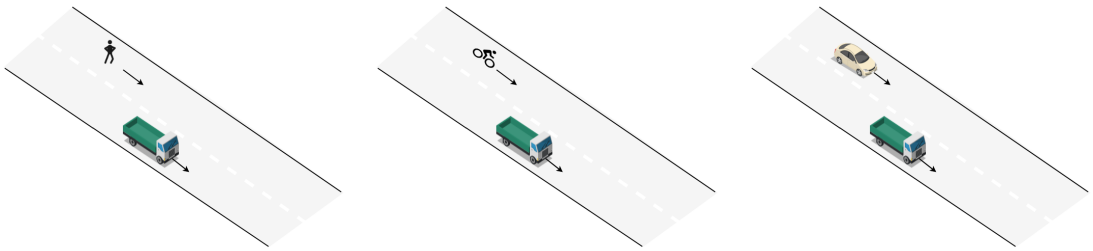


(a) Driving Scenario 8 pedestrian

(b) Driving Scenario 8 bicyclist

(c) Driving Scenario 8 car

Figure A.8: The target object and host vehicle accelerates to required speed. Host vehicle drives in reverse gear in 10kph. The speed of the car is 50kph, the speed of the bicyclist 30kph and the speed of the pedestrian 5kph.



(a) Driving Scenario 9 pedestrian

(b) Driving Scenario 9 bicyclist

(c) Driving Scenario 9 car

Figure A.9: The host vehicle accelerates to 40kph while the target is stationary in front of the host. When the host has driven past the target, the target accelerates to required speed. The speed of the car is 30kph, the speed of the bicyclist 30kph and the speed of the pedestrian 5kph.

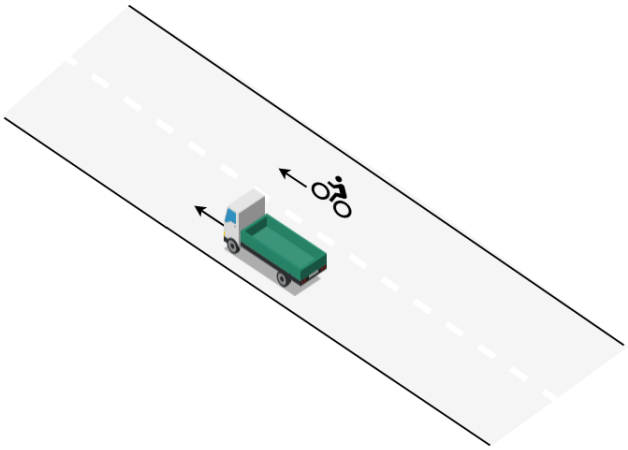
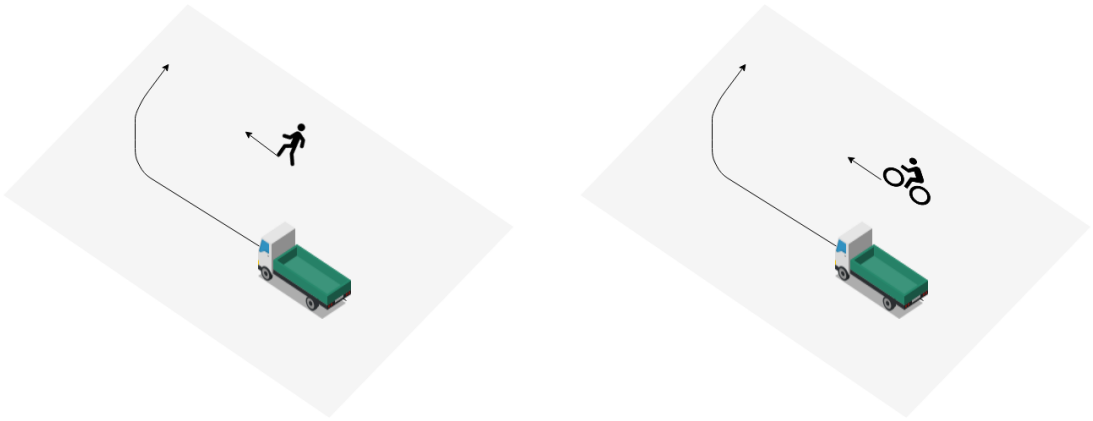


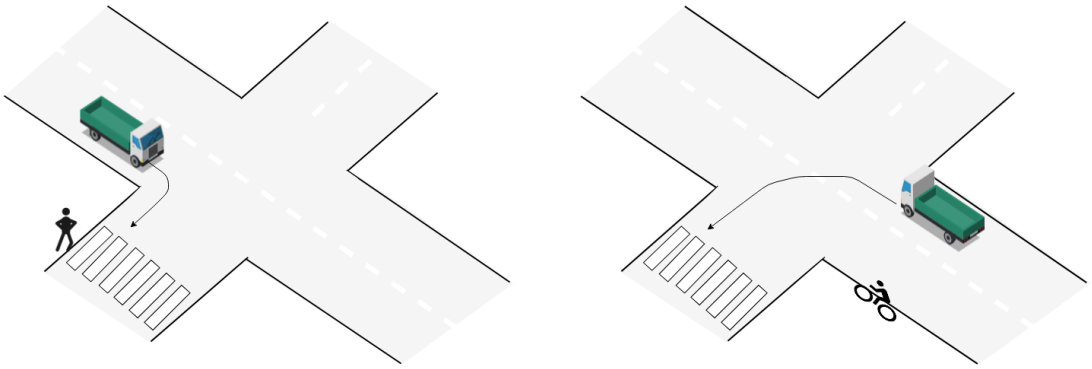
Figure A.10: Driving scenario 10. Both host vehicle and bicycle start by standing still next to each other. Both host and bicyclist then accelerates to 20kph. Scenario executed on both sides and in both directions.

A.3 Dataset T_2



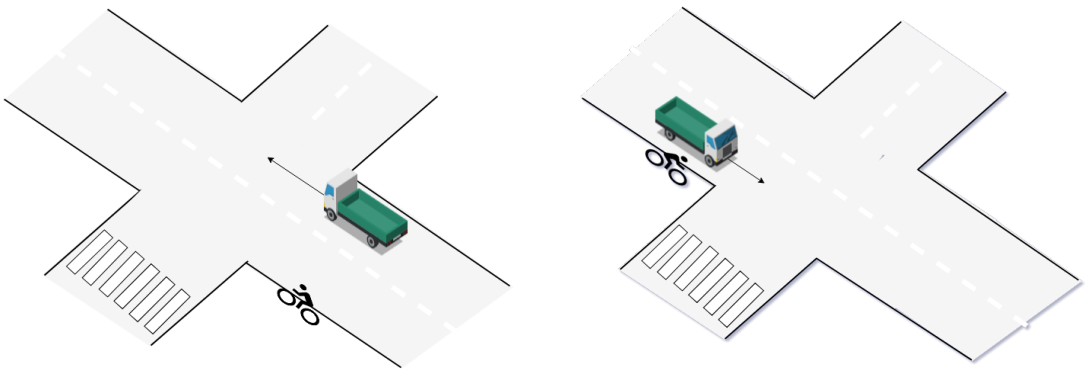
(a) Driving Scenario 11 bicyclist **(b)** Driving Scenario 11 car

Figure A.11: The target object and host vehicle accelerates to required speed. The target then turns in front of the target as illustrated. The speed of the host varies between 10kph, 15kph, and 20kph. The bicyclist speed and the pedestrians spe is 50kph, the speed of the bicyclist is 10kph and the pedestrians speed is 5kph. The scenario is executed with turns in both directions.



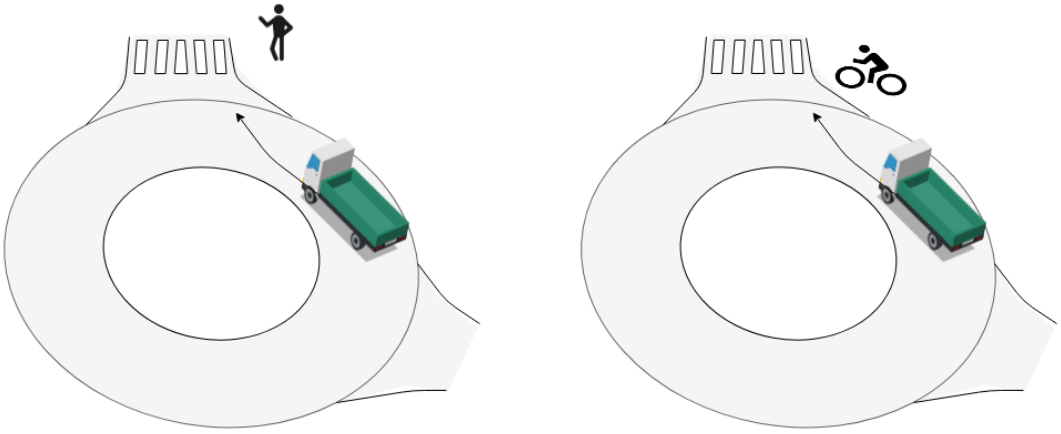
(a) Driving Scenario 12 pedestrian **(b)** Driving Scenario 12 bicyclist

Figure A.12: Host vehicle accelerates to 10kph and target to 20 kph (bicyclist) or 5 kph (pedestrian). Host vehicle then turn into pedestrian crossing or bicycle lane as is illustrated A.12a and A.12b. The driving scenario is executed with host driving in both directions.



(a) Driving Scenario 13 pedestrian **(b)** Driving Scenario 13 bicyclist

Figure A.13: Host vehicle accelerates to 30 kph and target to 20 kph. The driving scenario covers both when the target bicyclist is travelling in the adjacent lane to the host vehicle and when having a lane between the host vehicle and the bicyclist, as is shown in A.13a and A.13b.



(a) Driving Scenario 14 pedestrian **(b)** Driving Scenario 14 bicyclist

Figure A.14: Host vehicle accelerates to 5kph and target to 10 kph (bicyclist) or 5 kph (pedestrian). Both the host vehicle and the target travels through the roundabout. The host vehicle then turns out of the roundabout and hence into the pedestrian crossing as is illustrated A.14a and A.14b. directions.

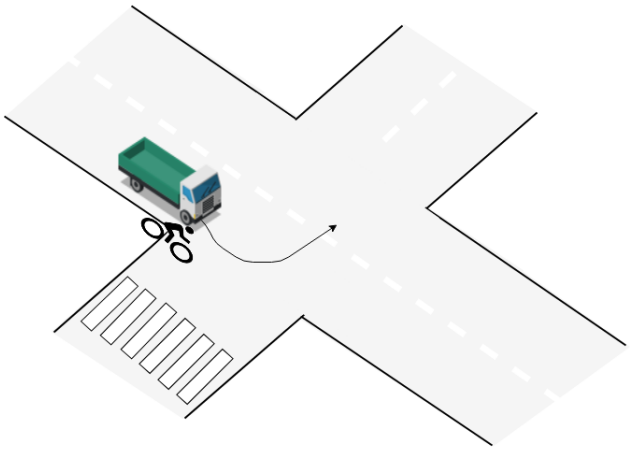


Figure A.15: Driving scenario 14. Host vehicle accelerates to 5kph and the target bicyclist to 20kph. In order to make the illustrated left turn the host vehicle makes a slight turn into the adjacent bicycle lane.

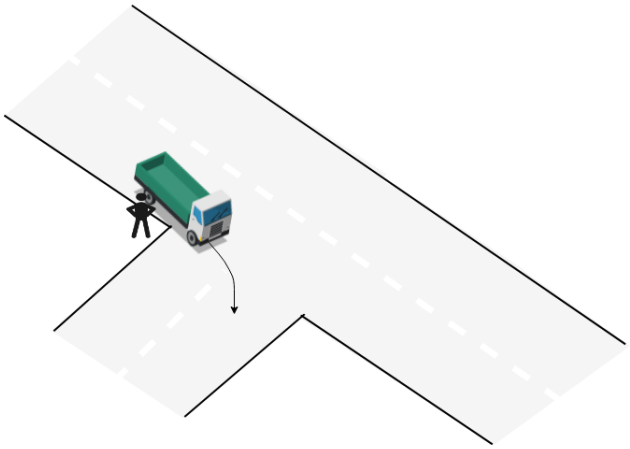


Figure A.16: Driving scenario 15. The host vehicle accelerates to 5kph and makes a tight turn which leads to the trailer cutting the sidewalk. The target is standing still on the sidewalk.