Moving Target Classification with Radar Point-Clouds and Supervised Contrastive Learning

Nils Zandler Andersson

Master's thesis
2021:E42

Lund University

Faculty of Science
Centre for Mathematical Sciences
Mathematical Statistics

Master’s Theses in Mathematical Sciences 2021:E42
ISSN 1404-6342

LUNFMS-3098-2021

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/

Abstract

This thesis deals with radar data for the purpose of moving target classification in the context of surveillance. The radar data in question comes in the form of point-clouds represented as frame-wise histograms with several channels and we seek to improve upon an existing cross-entropy based deep learning classifier using supervised contrastive loss.

We find that the embeddings output by supervised contrastive loss exhibit clearer separation between the classes and that it is possible to combine this approach with principal component analysis to obtain greater classification scores.

Acknowledgements

I would like to thank Arnab Bhattacharjee and Aras Papadelis for their persistent support, input and insight helping greatly in the writing of this thesis. I am also grateful to my supervisor Maria Sandsten for her good advice throughout the whole writing process.

Contents

Chapter 1

Introduction

Surveillance is the act of continuously observing someone or something in order to gather information. This could be in the interest of individuals, governments or private corporations and depending on the scenario, fully manual or fully automatic solutions may be appropriate. In any case, recent technological advancements in terms of computer hardware and algorithms have created entirely new opportunities in terms of what is possible to automate, with the possibility of replacing expensive human operators and in doing so freeing up human resources for other purposes.

Such automation does however come with the downside that trust needs to be transferred from human to machine so that the developers of the surveillance system need to make sure that it is reliable and does not behave unpredictably in practise.

If the purpose of the surveillance is classification, machine learning may be appropriate given that there is data available on which to train any suitable model. If enough labeled data is present, supervised deep learning models could be fitted and tested and this is the case in this work.

The provided data set has three classes: Human, Vehicle and Small. The former two are self-explanatory whereas the Small class consists mostly of animals but also some other types of targets.

We seek to improve upon an existing deep learning model (referred to as the baseline model) which is known to suffer from certain drawbacks such as a tendency to give false positives in the form of incorrect Human or Vehicle classifications which is a problem in cases where such false positives have a significant associated cost. Such false positives occur most frequently when animals are misclassified as humans due to similarity in the feature inputs.

Therefore an important metric of interest is the precision score for the respective classes as a measure of the models capacity to avoid such false positive results.

1.1 Problem Formulation

The cross-entropy cost function is the most well-known and commonly employed cost function in deep learning models for classification as of today. In spite of this, we believe it may be part of explaining the occurrence of some of the false positives mentioned in the previous section. This is because it has been observed to suffer from various drawbacks such as non-robustness to noisy labels and also the potential issue of poor margins in embedding space [1, 2].

These drawbacks may be particularly problematic where the application domain yields a different data distribution with respect to the one on which the deep learning model was originally trained, or where there are naturally occurring data corruptions, as the models ability to correctly embed the features that correspond to each class will then be degraded even though they may still be present in the received data stream. See Section 2.3.1 for a definition and description of cross-entropy loss.

The task of finding alternatives that persistently outperform cross-entropy loss at least under certain conditions is in general not a trivial task, as witnessed by its continued dominance in applications of deep learning.

By investigating the possibility of using supervised contrastive learning instead of cross-entropy based learning we may be able to overcome some of the aforementioned drawbacks of the cross-entropy approach which may account for the observed weaknesses of the provided baseline model. This attempt is inspired primarily by [3].

The results presented for the contrastive loss models are to be assessed in relation to those of the baseline model as it is this model we seek to improve upon.

1.2 What is Radar

Radar, like camera, monitors its surrounding environment using electromagnetic radiation. It is a technology which was originally developed in the first part of the twentieth century in connection to the second world war and the term radar came about as an acronym for radio detection and ranging [4].

Unlike camera however, radar both receives and transmits radiation through a set of receiving and transmitting antennas mounted on the system. In general, the frequencies produced by a radar are in the radio and microwave ranges making them invisible to the human eye and also able to penetrate some materials that visible light cannot.

The exact frequencies and emission patterns involved however are not fixed but vary from system to system depending on the particular use case in question. Likewise, the precise way in which the radar system processes the incoming information to generate useful data representations also varies.

The fact that radar needs to both produce and receive the radiation it relies on for surveillance presents a novel set of challenges and opportunities as compared with camera. Some advantages include

- robustness to adverse weather conditions
- ability to operate in darkness
- ability to surveil larger areas or at greater distances

Whereas some disadvantages may be

- unstructured data stream
- relatively low resolution
- inability to detect stationary targets

1.3 Frequency Modulated Radar

The type of radar underlying the data which this thesis builds upon is a so called frequency modulated continuous wave (FMCW) radar. This kind of radar transmits an outgoing signal with a given starting frequency that increases linearly with time and that lasts for some time. Each such signal is referred to as a chirp and a set of chirps together form a frame.

We measure the radial velocity by sending out multiple chirps. The received signal from all chirps will have the same phase if the object is stationary, but if it's moving, there will be small but detectable phase difference between each chirp. We perform a discrete Fourier transform to find the phase difference and deduce the velocity.

The (azimuth) angle is measured using multiple receive antennas, and the phase difference of a signal at each antenna is used to deduce the angle.

This framework allows the radar to also estimate the radial distance of the target, whereas if only the Doppler effect were to be exploited the radial distance would not be apparent. This presumes however that the target is non-stationary and not moving only tangentially to the radar.

In addition to detecting velocity and position, the radar also estimates the radar cross-section (RCS) of the target. This is a theoretical quantity that is essentially a measure of how detectable an object is by radar. Let σ stand for RCS, then

$$\sigma \sim (4\pi)^2 \frac{R^4 Q_r}{Q_t X} \tag{1.1}$$

where R (m) is the radial distance from radar to target, Q_t (W/m^2) is the power transmitted from the radar and where Q_r is the power density at the receiving antenna of the radar. X is a dimensionless quantity which depends on the antenna properties. See [5].

The R^4 factor in Eq. (1.1) is a consequence of the fact that the signal power reaching back at the radar is inversely proportional to the fourth power of the radial distance to the target and that the RCS values should not (at least theoretically) depend on the distance.

It turns out that the RCS value is one of the most useful quantities when it comes to classification of moving targets as it tends to differ greatly for different categories of targets. Vehicles made of metal for example tend to have comparatively high associated RCS as compared to animals, which may be useful when trying to distinguish between a rabbit and a car.

1.4 Radar Point-clouds

Point-cloud data is the basic unit of data used for classification in this thesis and how it is obtained is described in Section 3.1. It encodes the information described in the previous section across space and time for various identified targets in the form of point-cloud tracks. A track is represented by a number of points which are organized on the basis of which frame they correspond to where each frame is a time-slice of 100 milliseconds and are ordered chronologically. Associated to each point is a number of values such as its radial distance r to the radar, azimuth angle θ, radial velocity $\dot{r}$ and its RCS value rcs.

See Table 1.4 for an example of the first couple of frames of a track detected by radar. For a plot visualizing entire tracks for each of the classes, see Figure 1.1.

Table 1.1: Example data points from the first couple of frames of a track.

index	frame	r m	θ rad	$\dot{r}$ m/s	rcs m^2
1	0	14.4841	0.253540	0.45	0.004344
2	1	15.1941	0.259483	0.90	0.016569
3	1	15.1941	0.208720	0.30	0.000660
4	1	14.4841	0.254085	0.30	0.000547
5	2	15.1941	0.239290	1.05	0.013161
6	3	15.9031	0.248678	1.50	0.001250

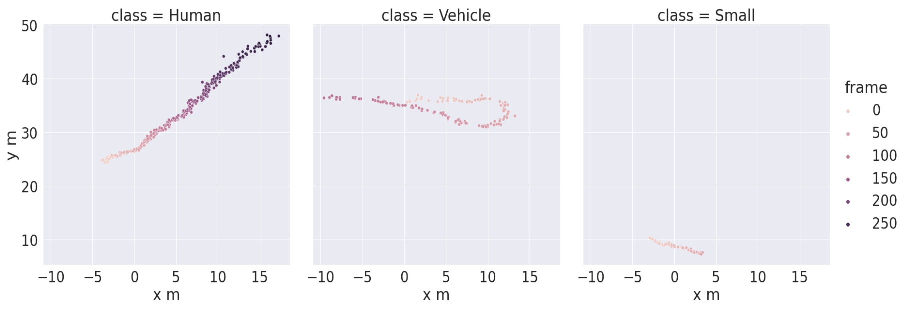

Figure 1.1: Example of point-cloud data tracks for each class. Each point represents the mean value of the x and y coordinates in a given frame. The plot illustrates the fact that Small class tracks tend to be relatively short.

1.5 Frame-wise Histograms

One way to create a structured representation of the point-clouds is to think in analogy to an image with a number of color channels. For a particular track, fixing each frame, we create a histogram for a number of selected variables. Considering p variables we obtain a linearly ordered sequence of p histograms per frame. This can be thought of as a 2D image with p color channels where the pixels correspond to bins and where their values are determined either by the number of points or some other statistic over the points in the corresponding bin. The channels considered in this thesis, in terms of the variables of Section (1.4), are

(1) $\dot{r}$ distribution in the range $[-128 \cdot 0.1243, 128 \cdot 0.1243]$

(1) mean centered r distribution

(3) mean rcs per bin

(4) linear rcs distribution in the range [0, 1]

(5) *rcs db* distribution in the range [-32, 32]

where every histogram has exactly 256 bins. Since channels (3)-(5) are trying to encode essentially the same information, each frame-wise histogram is assigned three channels where only one of the last three is included. Channel (3) is not really a histogram in the same sense as the others and it uses the bins of channel (1) in order to determine the points over which to calculate the mean RCS in each instance. The reason for not including θ is that the radar in general does not estimate this quantity with enough precision, leading to decreased resolution and thereby usefulness for classification. See Figure 1.2 for a visualization of the radial velocity distribution channel.

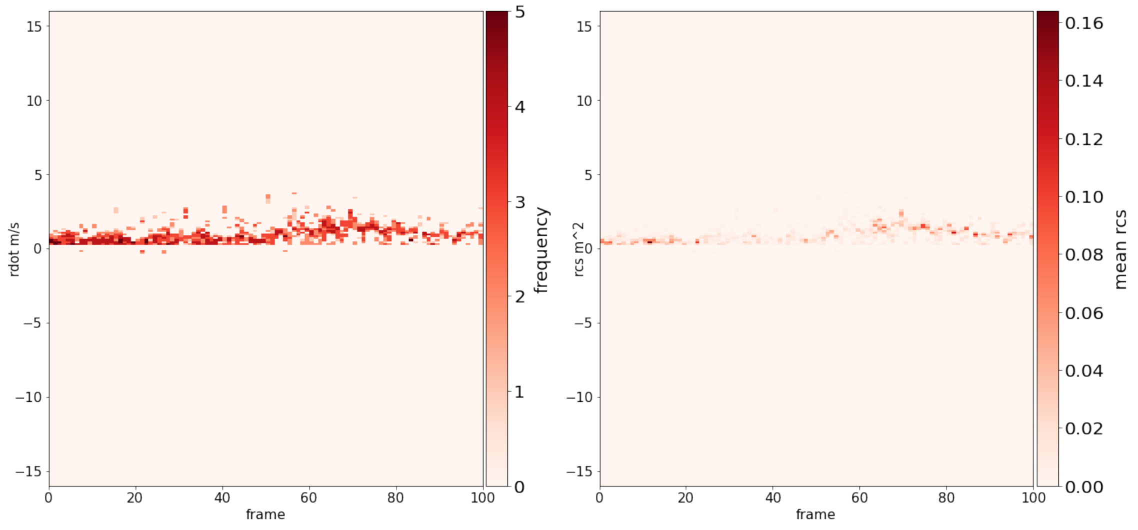

Figure 1.2: Visualization of the radial velocity and RCS channels of a frame-wise histogram representation of pointcloud data. On the horizontal axis are the frames and on the vertical axis are indices of the respective bins for the radial velocity. Each bin corresponds to a pixel and their values represent the number of instances of points in the corresponding bin or the mean RCS in the case of the right plot.

1.6 Work Outline

This sections aims to give an outline of the progression of this thesis. Chapter 2 provides the necessary conceptual pre-requisites needed to understand the models and methods used subsequently. The terms model and method are used somewhat interchangeably in this document however the word model has a more specific meaning in that it refers to an algorithm or a set of algorithms that can learn from data whereas the word method takes on a more general significance.

Chapter 3 presents the data used in the thesis along with some visualizations and statistics and it describes how it is prepared before training any model. Chapter 4 describes how the considered models are constituted at the basic level.

Chapter 5 is where the main methods and results of this work reside. At first, the methodology and setup behind supervised contrastive learning is presented. That is, how it works and how it is different from cross-entropy loss. Then the protocol which was employed to find a suitable model based on contrastive loss is described and a number of experiments are carried out in order to see if a superior model can be found. After these experiments have been carried out and their results presented, a new set of experiments are carried out with a novel methodology that we develop based on principal components (see Section 2.4.1) and K-nearest neighbors classification (see Section 2.5.2). Of course, results for these models are also presented. In the latter part of the chapter, we investigate some possible adaptations to the best discovered model, with relevance particularly relating to its application in a radar system for real time surveillance.

Lastly, Ch. 6 discusses the results of Ch. 5 and motivates some of the decisions that went in to the choice of experiments and investigations. It also addresses potential avenues for future work, including a novel point-cloud data representation which is presented in Appendix A.

Chapter 2

Preliminaries

Here we provide some necessary pre-requisites for the reader to follow along in subsequent chapters. First a set of definitions of relevant mathematical concepts is given, then the basic theory of deep learning is outlined including parameter estimation techniques and lastly, two dimensionality reduction methods useful for visualization of high dimensional data are described.

2.1 Mathematical Definitions

This section contains a listing of definitions of various mathematical concepts used in the subsequent sections of this document.

Definition 2.1.1 (1D Convolution)**.** Given two real-valued function f and g of a real argument, the convolution $f * g$ of f with g in $t \in \mathbb{R}$ is defined by

$$(f * g)(t) = \int_{\mathbb{R}} f(s)g(t-s)\ ds$$

and in the case of f and g being discrete functions the convolution of the two is defined as

$$(f * g)(t) = \sum_{k=-\infty}^{\infty} f(k)g(t-k).$$

Whenever the index of summation or variable of integration is excluded the sum or integral is implied from negative to positive infinity.

Definition 2.1.2 (2D Convolution)**.** Given discrete functions I and K of two integers, the 2D convolution S of I with K is given by

$$S(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

where summation is computed over all combinations of m and n where the image has non-zero corresponding pixel values, or equivalently over all of $\mathbb{Z}^2$. K is referred to as the kernel and S is the feature map.

Definition 2.1.3 (Sigmoid function)**.** For a real variable x the Sigmoid function σ is defined by

$$\sigma(x) = \frac{e^x}{1+e^x}.$$

Definition 2.1.4 (Softmax function)**.** Given a set of real variables $x_1, x_2, \ldots, x_n$, we define the Softmax function S by

$$S(x_1, x_2, \ldots, x_n) = \frac{1}{\sum_{k=1}^n e^{x_k}}(e^{x_1}, e^{x_2}, \ldots, e^{x_n}).$$

Definition 2.1.5 (Rectified linear unit)**.** Given a real variable x, we define the rectified linear unit Relu as

$$\mathrm{Relu}(x) = \max(x, 0).$$

Definition 2.1.6 (L1 and L2 Norms)**.** Let $x \in \mathbb{R}^n$ where n is a positive integer. The L^1 norm of x is defined by

$$||x||_{L^1} = \sum_{k=1}^n |x_k|$$

and the L^2-norm of x is given by

$$||x||_{L^2} = \left(\sum_{k=1}^n x_k^2\right)^{\frac{1}{2}}.$$

Definition 2.1.7 (Precision and Recall)**.** Given a classifier we define for each respective class

$$\text{Precision} = \frac{tp}{tp+fp} \quad \text{and}$$
$$\text{Recall} = \frac{tp}{tp+fn}$$

where tp, fp and fn stand for true positives, false positives and false negatives respectively.

Definition 2.1.8 (Accuracy and F-score)**.** Given a classifier, the accuracy score is defined by

$$\text{Accuracy} = \frac{\text{p}}{N}$$

where p is the number of correct predictions and N is the total number of prediction made. The F-score on the other hand is defined on the basis of the concepts of Definition 2.1.7 and is given by

$$\text{F-score} = \frac{2pr}{p+r}$$

where p is the precision score and r is the recall score.

2.2 Machine Learning Basics

Here we provide a brief overview of some basic concepts and ideas in machine learning that are essential for the sections to come. It is based primarily on chapters 5, 6, 9 and 10 of [6] and the reader is referred there for more detail.

2.2.1 The Notion of Machine Learning

Machine learning (abbreviated ML) is a discipline which aims at the development of algorithms that can automatically adjust a collection of state variables or parameters to become more able at a particular processing task. Usually this process of automatic adaptation occurs by providing the algorithm with a set of training data points that may be processed either all at once or in smaller packets called batches. In this context, a typical protocol is to split the available data into two separate sets, one for training and one for testing. In this work, these are referred to as training and test set respectively and the training set may be further split into a so called validation set.

The term machine learning is today a very broad term which encompasses a wide range of techniques and applications in alignment with the aforementioned description and the field can be divided into several categories. This thesis primarily deals with the sub-field that is supervised learning for the purpose of classification. In particular the classification of moving targets as surveilled by a radar system.

2.2.2 Classical Machine Learning Techniques

Perhaps the most canonical and fundamental machine learning algorithm for classification is logistic regression of a binary outcome variable $Y \in \{0, 1\}$. Given m predictor variables $x_1, \ldots, x_m$ and real parameters $\beta_0, \beta_1, \ldots, \beta_m$ one computes the probability p that $Y = 1$ via the relation

$$\log \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m \tag{2.1}$$

which is equivalent to

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m)}} \tag{2.2}$$

where log refers to the natural logarithm. The graph of the function implicated in Eq. (2.2) is the sigmoid function which is defined in Definition 2.1.3 and displayed in Figure 2.1 along with its derivative.

The β parameters are usually determined with the training set using maximum likelihood estimation (MLE) carried out numerically. This turns out to be equivalent to minimizing the cross-entropy between the empirical distribution of the training set and the model distribution given the parameters and relates to the customary cross-entropy loss function used in the context of deep learning discussed below.

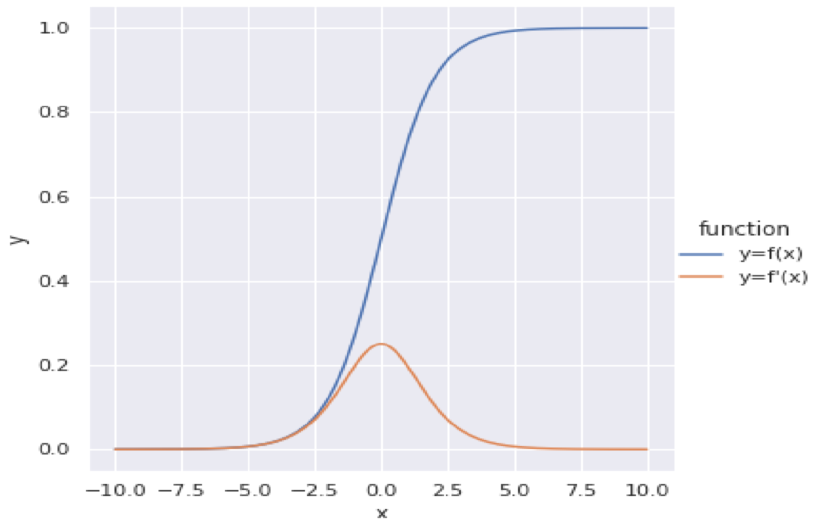

Figure 2.1: Graph of the sigmoid function $y = f(x)$ in blue and its derivative in orange. It is the fundamental function underlying the softmax function for classification and is equivalent to the softmax function if the problem is binary.

2.2.3 Artificial Neural Networks

An artificial neural network (subsequently referred to as simply neural network) is a kind of function mapping between predictors and response variables based on function composition which depends on a set of parameters θ that are adjusted during training.

The name neural network derives from the fact that these models where originally intended as computational models for biological brains (such as the mammalian one). In spite of this history, they are not to be thought of as equivalent computational systems mimicking the function of biological brains as they are not generally regarded as realistic models of how biological systems function (see chapter 1 of [6]).

Each function application in the composition can be thought of as generating a new representation for the feature data and the idea is that each successive representation obtained should be more useful when it comes to predicting the response variable. Each such representation is referred to as an embedding and in this work we are primarily interested in the final embeddings output by each model.

Neural networks can be used for both regression and classification and their most basic building blocks are as follows:

(i) Hidden and visible layers

(ii) Weights and biases

(iii) Activation functions

In the context of classification, what is visible to us are the values of the predictor variables and the outgoing probabilities estimated by the model for each class. In between these values are the so called hidden layers consisting of units referred to as neurons. Each unit is a real number which is determined by the values of other units to which it is connected depending on the current state of the parameter vector θ and the exact nature of the network. The parameters of the model are referred to as weights and biases, where the weights determine the degree to which the value of a particular unit influences the values of connected units and where the biases are intercept terms adding constant values to each unit for each connection. Furthermore, associated to each hidden unit is an activation function (which may be the identify function) that applies a potentially non-linear transformation to a linear combination of the weights, biases and values of connected units. A common choice of activation function is the rectified linear unit (Relu) defined in Definition 2.1.5. See Figure 2.2 for a visual representation of a simple neural network.

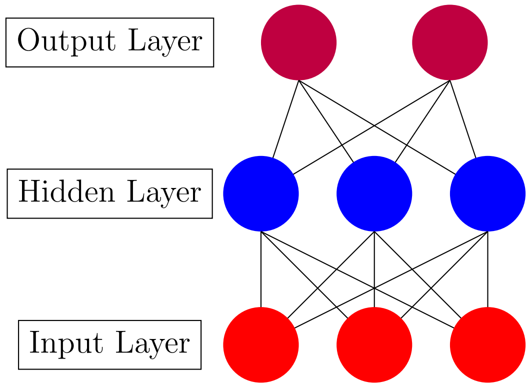

Figure 2.2: Example of a simple artificial neural network called a perceptron. In this case there are three densely connected layers of which one is "hidden". The upper two units could represent probabilities for two classes or be real numbers regressing two quantitative outputs.

2.2.4 Convolutional Neural Networks

In this work two more advanced types of networks are employed, namely convolutional and recurrent networks.

Convolutional networks consist of convolutional layers and are primarily used for the processing of two dimensional image data that may have a single or several color channels. Just like the original neural networks, convolutional networks were originally inspired by biology, in particular by how the mammalian visual system functions (see page 15 in [6]). A convolutional neural network has the following main elements in addition to those common to all neural networks:

(i) Kernel

(ii) Convolutional layer

(iii) Pooling layer

In the case of one dimensional inputs, the kernel is a one dimensional, discrete function and corresponds to the function g in Definition 2.1.1 and the input is represented as the values assumed by the function f. When the input is two dimensional, it builds on the same pattern but generalized to two dimensions. If I is a function of two integers which represents for example an image, then the convolution of I with the kernel K is given as in Definition 2.1.2.

A convolutional layer is where this convolution is computed. Since the function S

depends on the same number of variables as the input, it retains the spatial structure of the input. If the input is for example and image, the output can also be thought of as such.

On each value of the feature map an activation function such as the Relu function is computed in a detection stage.

Pooling entails computing a summary statistic over a set of subsets of the outputs from the detection stage. Usually these subsets are hyper-cubes of the same dimensionality as the original input and the summary statistic to be computed may be the maximum or mean values in each such hyper-cube. The pooling operation reduces the dimensionality of the output and allows for feature detection. See Figure 2.3 for a visual depiction of a convolutional neural network.

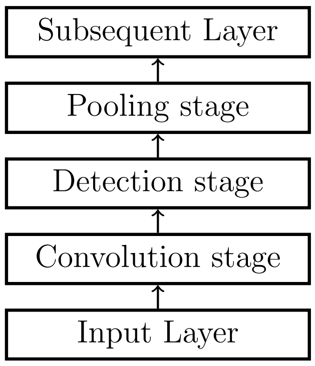

Figure 2.3: Flowchart of a convolutional network. At first, the inputs are convolved with the kernel and the output feature map goes on to a detection stage where an activation function is computed on each value. Pooling is then applied and the output is fed into the subsequent layer. The middle three stages can be thought of as a single convolutional layer.

2.2.5 Recurrent Neural Networks

Recurrent neural networks (abbreviated RNN) is a family of neural networks specialized for the processing of sequential data. Just like CNNs implement parameter sharing across space, RNNs implement parameter sharing through time. Even though a one-dimensional convolutional network can also act on sequential data, if it is to implement significant parameter sharing, it can only do so shallowly as any particular convolution output will only depend on a fixed number of contiguous values in the input. An RNN overcomes this limitation by repeated function composition.

Let $f = f(x|\theta)$ denote a vector-valued function of a vector x where θ is a set of parameters, $h^{(t)}$ the hidden units at time t, $x^{(t)}$ the inputs to the model at time t. Then in its simplest form an RNN can be defined by

$$h^{(t+1)} = f(h^{(t)}, x^{(t)}|\theta).$$

In this work we employ models that involve both CNN and RNN layers.

2.2.6 Model Capacity and Typical Trade-offs

When we speak of a model's capacity (sometimes also called flexibility) we are referring to the model's ability to represent various relationships between the predictors and outputs. Usually a model's capacity will increase with the number of parameters and thereby also its ability to fit well the provided data set. However, since the goal of machine learning is not so much to fit the training data set well as it is to provide accurate predictions for future observations, high accuracy on the training set is no guarantee for low generalization error. There are two essential concepts that qualitatively describe a models capacity, namely under-fitting and over-fitting.

Over-fitting occurs when the model capacity is significantly larger than what is actually required to capture the relationship between the inputs and the outputs as available in the training data. In such a scenario the model begins to fit patterns that are specific to the given training data set but that are not fundamental to the true relationship between the inputs and the outputs. This results in comparatively low error rates on the training set, but when new data is introduced the fitted patterns instead prevent the model from yielding accurate predictions. In general, over-fitting tends to become less of a problem as the size of the training data set increases.

Under-fitting on the other hand occurs when the model capacity is significantly lower than what is actually required to capture the relationship between the inputs and outputs. In this case, the model will not tend to fit the specific patterns of the training data set but will still be unable to make good predictions for future observations.

It can be shown that

$$\text{MSE} = \mathbb{E}[(\hat{\theta} - \theta)^2] = \text{Bias}(\hat{\theta})^2 + \text{Var}(\hat{\theta}) \tag{2.3}$$

where MSE is the mean squared error, θ is the true parameter vector, $\hat{\theta}$ is the estimated parameter vector and where $\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$.

Equation (2.3) is a bias variance decomposition of the mean squared error which can be interpreted in view of the concepts of over and under-fitting. Models that under-fit tend to exhibit low variance but high bias and models that over-fit tend to exhibit low bias and high variance. A good example of a low capacity model which tends to show low variance is the linear logistic regression model of Eq. (2.1).

Another well known trade-off in the context of machine learning is the trade-off between accuracy and interpretability. Modern neural networks are in a sense black boxes whose inner representations are unknown to the practitioner while they are capable of representing a rich array of possible relationships. The logistic regression model of Eq. (2.1) is instead not very accurate when it comes to non-linear data while it is highly interpretable.

2.3 Estimation Techniques

This section gives a brief overview of the theory behind fitting the parameters of a deep learning model, including optimization, regularization and hyper-parameter selection.

2.3.1 Network Optimization

In the the context of neural networks, optimization is referred to as learning. It differs from other forms of optimization problems in that one seeks to maximize or minimize the true objective P (for example expected prediction accuracy) only indirectly through minimizing a cost function J (also called objective function). This has to do with the fact that neural networks can potentially have thousands or hundreds of thousands of parameters and that direct optimization of the true objective would be intractable. We can think of the objective function in two different ways:

$$J(\theta) = \mathbb{E}_{(x,y)\sim\hat{p}_{data}} L\big(\hat{y}(x,\theta), y\big) \quad \text{or alternatively} \tag{2.4}$$

$$J^*(\theta) = \mathbb{E}_{(x,y)\sim p_{data}} L\big(\hat{y}(x,\theta), y\big) \tag{2.5}$$

where the former expectation is taken over the empirical distribution of the training set, the latter is taken over the true data generating distribution and where L is the per example loss value. Ideally we would like to minimize the latter but since we do not know the true data generating distribution we instead are forced to optimize on the basis of the former.

The loss function used in equations (2.4)-(2.5) could be simply the one-zero loss where the value zero is assumed in the case that the prediction $\hat{y}(x,\theta)$ is correct and one otherwise. However this loss is not differentiable and does therefore not allow for stochastic gradient descent (SGD) to be applied which is why we instead use the negative log-likelihood of the prediction, which is known as cross-entropy loss. Mathematically,

$$L\big(\hat{y}(x,\theta),y\big) = -\log p^{(i)} \tag{2.6}$$

where $p^{(i)}$ is the probability of the correct class predicted by the model for a particular data point with features x and label y.

Given the training data set and initial parameter vector θ_0, stochastic gradient descent is used to iteratively update the parameter vector θ. This is done by splitting the training set into a number of batches of equal size and changing the vector θ by taking a step in the direction of the negative gradient of the cost function as computed over that batch. The step is scaled by a parameter called the learning rate, which is a parameter that can be tuned on during hyper-parameter selection, just like the batch size used.

For more information relating to network optimization, see chapter 8 of [6] .

2.3.2 Network Regularization

Regularization entails imposing some restriction on the model parameters or training process in order to obtain better generalization. There are several ways in which this can be done and frequently one sacrifices some accuracy on the training set for better accuracy on the test set. This relates closely to the trade-off illustrated by Eq. (2.3) as regularization tends to increase the model bias for the sake of reducing its variability thus reducing generalization error if the decrease in variance out-weighs the corresponding increase in bias. This is especially useful in the case of deep learning where model capacity may greatly surpass what is actually required to fit the data well, leading to over-fitting.

A common form of regularization is based on penalizing the involved model parameters. In this case one seeks instead to minimize

$$\tilde{J}(\theta) = J(\theta) + \alpha\Omega(\theta) \tag{2.7}$$

where α is a non-negative hyper-parameter and where Ω imposes a penalty value on the parameter vector θ. When $\alpha = 0$, the regularized objective of Eq. (2.7) is equivalent to its non-regularized counterpart and as α increases, more and more regularization is applied leading to the ultimate case of $\alpha = \infty$ in which the parameter search space is maximally restricted.

A common variant of parameter penalization is based on the L^2 norm of the parameter vector where

$$\tilde{J}(\theta) = J(\theta) + \alpha\theta^T\theta. \tag{2.8}$$

As an alternative to norm penalties, there exists a form of regularizing of neural networks known as dropout. Dropout works not by imposing a penalty on the parameters according to some function but rather by cancelling, that is setting to zero, the values of a random fraction of the units in a specified layer according to a specified rate.

If the rate is 0.5 and dropout is set for a particular hidden layer in the network, then on average half of the units will be set to zero in each forward pass during training. This has the effect of reducing the prevalence of haphazard train data specific patterns that correspond to over-fitting with the result of better generalization.

For a more thorough description of these regularization techniques, see chapter 7 of [6].

2.3.3 Hyper-Parameter Selection

Every model is specified by a number of parameters referred to as hyper-parameters. These parameters are referred to as such since they are not adjusted during training on the training set. Once the hyper-parameters are selected, the model is fixed and its eventual performance when fitted to the entire training set can only be affected by randomness related to the initialization of its parameters (unless the random seed is also treated as a hyper-parameter which is recorded).

We can assign a score to each set of hyper-parameters in order to be able to discriminate between models and choose the best one. In this thesis this is done using 5-fold cross-validated accuracy on the training set. When a number of models have been assessed, the best one is selected and tried on the test set.

In addition to specifying the model itself, the hyper-parameters also determine the training process. For example it includes the learning rate, batch size and number of training epochs.

2.4 Dimensionality Reduction Methods

When working with data with more than four dimensions, it is impossible to visualize it all directly without reducing it in some way first. Therefore, we are in need of various methods that allow for such reduction that allow for visualization (and modelling) in one to three dimension. Two useful methods to this end are principal components and t-distributed stochastic neighbor embedding.

2.4.1 Principal Component Analysis

Principal component analysis (PCA) accomplishes dimensionality reduction by first finding the principal components of the data set. These are essentially the directions along which maximal variability occurs and can be ordered on this basis. In general there are as many principal components as there are dimensions in the original data set under consideration and it can be shown that they are given by the eigen-vectors of the data covariance matrix with corresponding eigen-values determining their relative significance in terms of explaining the variability in the data set. Since any covariance matrix needs to be both positive-definite and symmetric, the eigen-values will all be positive and the corresponding eigen-vectors can be used to construct an orthonormal basis.

Once the (orthonormal) principal components have been computed, one to three of them with the greatest eigen-values are selected and a projection matrix is constructed which can be applied to any point of the original data set. Projecting all the points onto 1D, 2D or 3D space yields a representation which can be visualized and which preserves as much of the variability as is explained by the selected principal components.

The effectiveness of PCA for visualization is intrinsically predicated upon the variability of the data being asymmetrically distributed in high-dimensional space. In the case that it is not, that is to say it is spread evenly across each dimension, no matter which principal components are selected, if one is to select just a few of them, only a small fraction of the total variability will be captured by the resulting projection.

The main advantages of PCA however is that it is easily reproducible given the same data and that it is a linear method which can be intuitive to interpret. For a simple example of principal components, see Figure 2.4.

2.4.2 t-Distributed Stochastic Neighbor Embedding

t-distributed stochastic neighbor embedding (t-SNE) is, unlike PCA, inherently a non-linear method. It is also a probabilistic approach, meaning that one may get different results even for the same exact data unless the random seed is recorded and re-used.

It has two main stages. In the first stage a probability distribution over pairs of points is constructed in such a way that similar objects (as measured for example by euclidean distance) are assigned high probabilities whereas dissimilar objects are assigned low probabilities. It then constructs a corresponding probability distribution in low dimensional space and minimizes the Kullback–Leibler divergence between the two distributions using gradient descent.

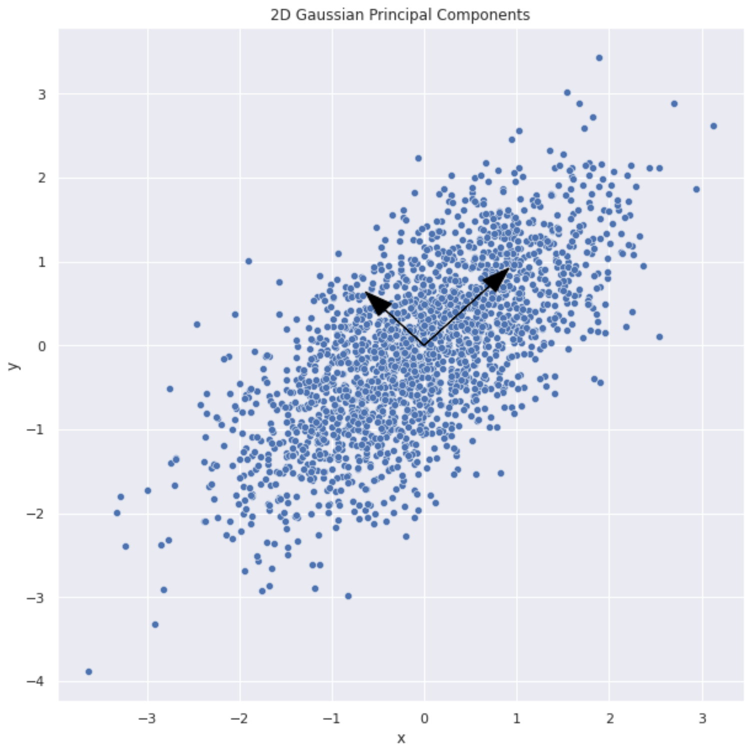

Figure 2.4: Principal components of 2000 points simulated from a 2D multivariate normal distribution with mean at the origin, unit variance and covariance 0.7. The principal component pointing in the top right direction corresponds to the largest eigenvalue and captures the majority of the variability present in the data. The plotted vectors have been scaled for illustrative purposes.

The result is a map between the original space and a low dimensional space which tends to preserve clusters, although the outcome can be significantly dependent upon choice of parameters. For a more thorough description of the method see [7].

2.5 Two Specific Classification Methods

This section presents two additional methods which can be used for classification: Random forest classifiers and K-nearest neighbors.

2.5.1 Random Forests

A random forest classifier is an ensemble method which is based on aggregating the predictions made by a number of individual classification or regression trees. See Figure 2.5 for a schematic outline of such a tree. In the case of classification, a random forest entails fitting N classification trees according to an algorithm which yields unique trees in each instance. When given a new data point, each tree then produces a class prediction and the majority vote is chosen as the final classification.

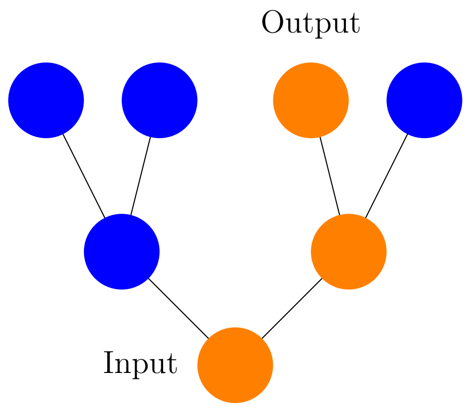

Figure 2.5: Simple schematic of a classification or regression tree. The input features enter the orange bottom node and the subsequent node is chosen on the basis of a heuristic condition. At the end, each node corresponds to a particular value or class and a unique one is arrived at.

2.5.2 K-Nearest Neighbors

The K-nearest neighbor algorithm is a non-parametric method for classification or regression. It is based on a very simple premise, namely that there is a similarity metric available and that the final prediction for an input data point depends on the values of the k (which needs to be chosen in some way) nearest data points in the training set as determined by the similarity metric. Usually the metric used for computing similarity is simply the Euclidean distance between the input features.

In the case of classification, the class belongings of the k nearest neighbors are computed and the majority vote is taken as the final prediction for the input.

Chapter 3

Data

In this chapter we give a description of the point-cloud data that each model is implicitly trained on and a structured representations that each ML model can be trained on directly.

3.1 Radar Processing Pipeline

In order to use radar for classification there needs to be a processing scheme or system in place which can somehow make sense of the information coming back to the radar in the form of electromagnetic radiation.

At first, the radar registers the incoming data and converts it via an analog to digital converter (ADC) to electronic format suitable for processing by a computer. It then performs a pre-processing step followed by three phases of fast Fourier transforms (FFT) that extract the range, velocity and angle of arrival respectively. After detection (thresholding), this outputs a set of points referred to as a point-cloud. It proceeds to do clustering and tracking which gives a more structured point-cloud representation of a track that allows for the tracing of a target over time. See [8] for a more thorough description and derivation of these steps.

The length of tracks is variable and the number of points per frame (which serves as a unit of time) also varies. This creates a data stream which is unstructured in relation to the data stream generated by camera, which is why an intermediary representation between the radar point-clouds and the actual classifier is used.

From the point of view of this thesis the point-cloud tracks are the most basic unit of data and we do not concern ourselves explicitly with the steps which precede it. Since the processing is often done on embedded devices, which are usually less powerful than stationary desktop computers with access to modern graphics cards, there are hardware constraints which implicitly guide our choice of machine learning models. That is to say, if there are two models that perform similarly well, and one of them is much cheaper computationally, there is a strong incentive to prefer it over the other. It is worth mentioning however that the actual training of the models is not done on an embedded device and so heavy computational cost in this respect is less of a concern. See Figure 3.1 for a flow-chart outlining the radar processing pipeline as described above.

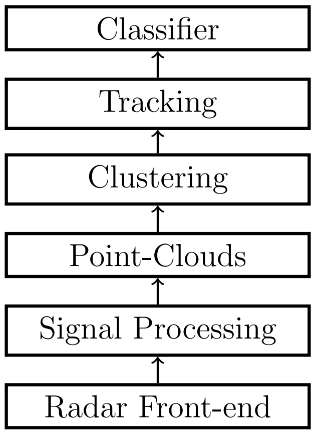

Figure 3.1: Simplified flow-chart of how radar for moving target classification may process the incoming data stream. At first, electromagnetic radiation reaches the receiving antenna of the radar. The signal is then converted to digital format and processed by an embedded computer in several steps till a data representation suitable for classification is obtained.

3.2 Data Visualization and Statistics

This section presents statistics and plots that help to illustrate some of the properties and challenges of the utilized training and testing data sets of point-cloud tracks. Tables 3.1-3.2 present some basic facts. Table 3.1 reveals that in both the case of the train and test set, the Small class has by far the fewest number of points. This is true in spite of the fact that the Small class has the most amount of tracks in the train set, which is due to these tracks being shorter on average and having fewer points per frame.

There are some outliers in the data set such as the Human point observation in Table

3.2 with a radial velocity of 18.75 or the Vehicle point observation in Table 3.3 with an RCS value of almost 800. Even though these tracks could have been removed from the train data set, the prevalence of such outliers is sufficiently rare as to be unlikely to significantly affect the final results anyhow considering that it consists of over a million points.

Figures 3.2 and 3.3 contain histograms with logarithmic scale of track-wise summary statistics of the absolute radial velocity and RCS values in the train data set depending on class.

Table 3.1: Some basic facts about the data sets used for training and testing. In both cases of the train and test set Small has the fewest number of total data points.

	Human	Vehicle	Small
Train Tracks	804	430	1307
Train Points	1 167 772	529 317	359 637
Test Tracks	377	326	315
Test Points	684 315	768 238	131 735

Table 3.2: Summary statistics of the absolute radial velocity over the points in the training data set depending on class including the 25, 50 and 75 percent quantiles.

	mean	std	min	25%	50%	75%	max
Human	1.26	1.02	0.25	0.62	0.99	1.50	18.75
Vehicle	3.30	2.58	0.25	1.05	2.70	4.97	15.41
Small	1.11	1.11	0.25	0.37	0.75	1.49	13.55

Table 3.3: Summary statistics of the RCS value over the points in the training data set depending on class including the 25, 50 and 75 percent quantiles.

	mean	std	min	25%	50%	75%	max
Human	0.05	0.11	0.00	0.00	0.02	0.05	28.98
Vehicle	0.38	4.34	0.00	0.02	0.05	0.15	788.12
Small	0.02	0.05	0.00	0.00	0.01	0.02	3.95

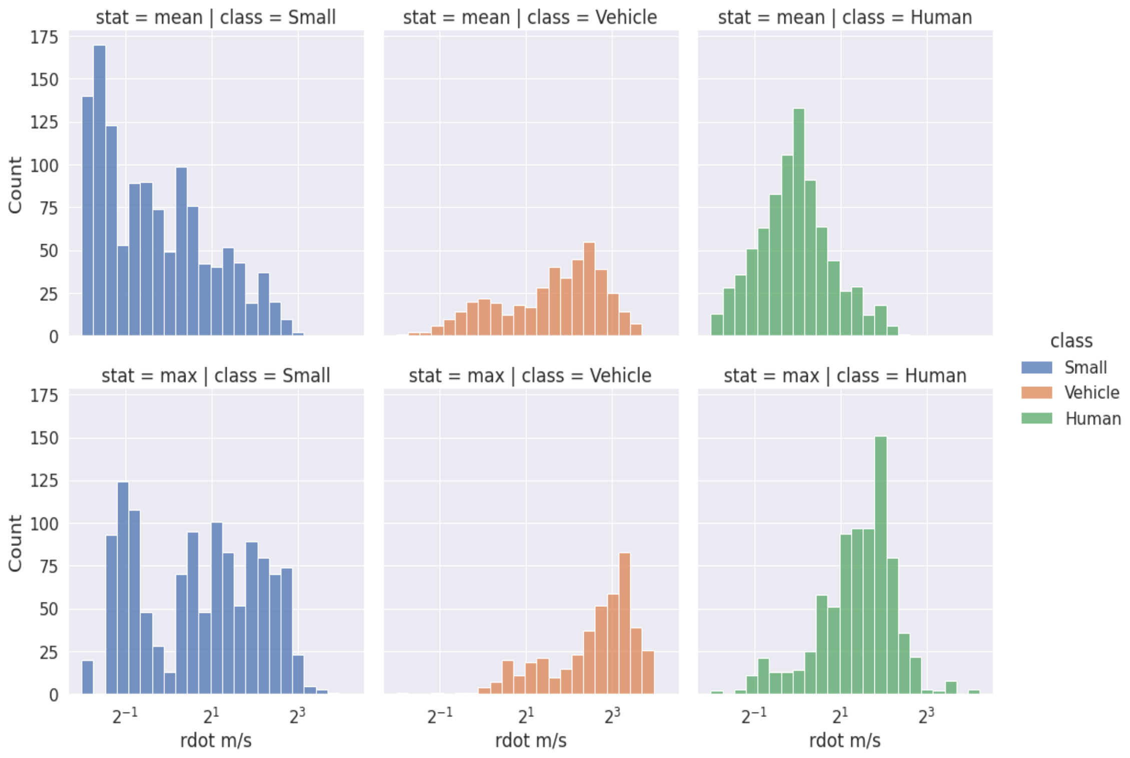

Figure 3.2: Track-wise summary statistics computed over the train set for each respective class with absolute radial velocity. The rows determine the summary statistic computed and the column the class. The scale is logarithmic, as the distributions are heavily skewed in the linear domain.

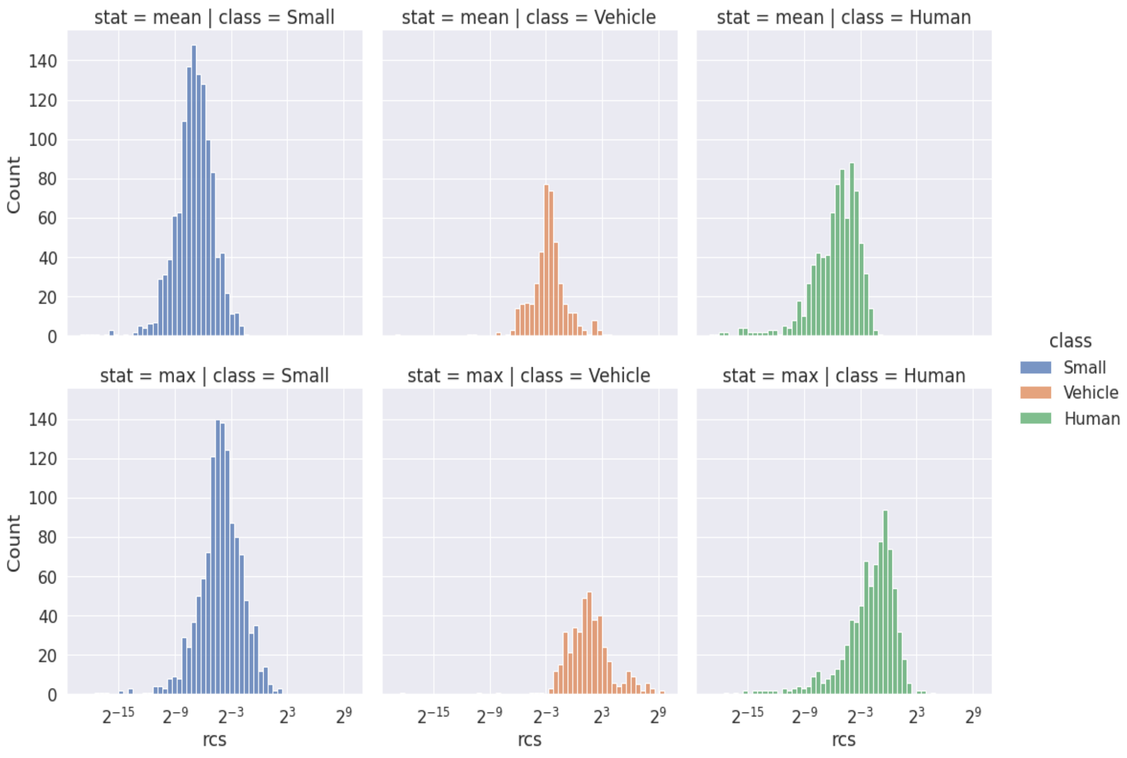

Figure 3.3: Track-wise summary statistics computed over the train set for each respective class with RCS. The rows determine the summary statistic computed and the column the class. The scale is logarithmic, as the distributions are heavily skewed in the linear domain.

3.3 Windowing

Since each track has its own unique length and we seek a structured representation, a number of windows of fixed length are obtained from each track. The window size is not given and has to be chosen with the use-case in mind since it affects the computational load and time for each classification. We treat it as a hyper-parameter with the two values 20 and 21. The main reason for not allowing a greater variety of window sizes is to limit the total number of combinations in the search space.

These windows are then broken into a number of segments of length either 5 or 7 as to divide the length of the original window and the segments are processed individually by a time-distributed convolution.

3.4 Data Augmentation

There are several ways in which one could potentially augment the data. Either one could do it on the level of the point-clouds themselves or on the level of a point-cloud representation such as the frame-wise histograms. We do augmentation on the level of the representation and consider the following techniques:

- Time-extension: Tracks that are too short (less than 21 frames) but longer than 11 frames are extended by appending the ultimate frames in reverse order so that a length of 21 frames is attained.
- Frame-flip: Each track generates another which plays itself out in temporally reversed order.
- Bin-flip: Each track generates another by reversing the order of the bins.

The time-extension augmentation in particular allows us to extract some value even from the tracks that otherwise would have been to short given the need for a fixed window size, enabling use of a greater fraction of the data.

Chapter 4

Model Description

This chapter describes the TPOT framework for finding ML classifiers and the basic neural architecture that underpins each model of this thesis which are all based on the frame-wise histograms presented in 1.5.

Also presented is the baseline model which is used for comparison with the obtained models and the range of hyper-parameters which specify the model within the constraints of the outlined architecture are presented as well.

4.1 TPOT AutoML framework

The tree based pipeline optimization tool (TPOT) is a module of the Python programming language for automatic machine learning presented in [9]. It is capable of automatically discovering pipelines which may be suitable for a given dataset using cross-validation and genetic programming. It tries models such as logistic regression, random forests and gradient boosting and it employs various feature pre-processing and selection methods. See [9] for a more thorough description.

4.2 Frame-wise Histogram Based Models

For the frame-wise histogram representation of the point cloud data a model architecture based on a CNN component followed by an RNN component is employed. The CNN component is time distributed across its segments and consists of one or two layers, has kernel size two or three and may have a number of filters in a given range. The RNN component then takes the flattened output of the CNN and generates a set of embeddings via sequential processing either on the basis of cross-entropy loss or supervised contrastive loss. In the case of cross-entropy, the embeddings are generated in unison with the final classification layer consisting of three densely connected units with softmax activation function which affect the embeddings output from the RNN layer. In the case of supervised contrastive loss, the embeddings are generated independently of any final classification layer and they are the outputs of the RNN layer normalized to the unit sphere in the same space.

4.3 Baseline Model

In order to make sense out of the models and scores presented in Section 5 it is useful to have a baseline model in mind. This is a frame-wise histogram based cross-entropy model which makes predictions in alignment with the description given in Section 4.2. It attained an accuracy of 84.90% and Table 4.1 displays the recall and precision scores when computed over the first 21 frames in the testing set. Additionally, Figure 4.1 contains a t-SNE plot of the corresponding embeddings as computed over the train set.

The figure reveals poor separation between the Human and Small classes which may be part of explaining the models tendency to give false positives in the case of input feature corruption. It does however not explain why the model has such a poor recall score for the small class as opposed to the human class, that is why it tends to favor Human over Small.

Table 4.1: Recall and Precision scores for a cross-entropy based baseline model

	Human	Vehicle	Small
Recall	0.86	0.94	0.49
Precision	0.83	0.91	0.70

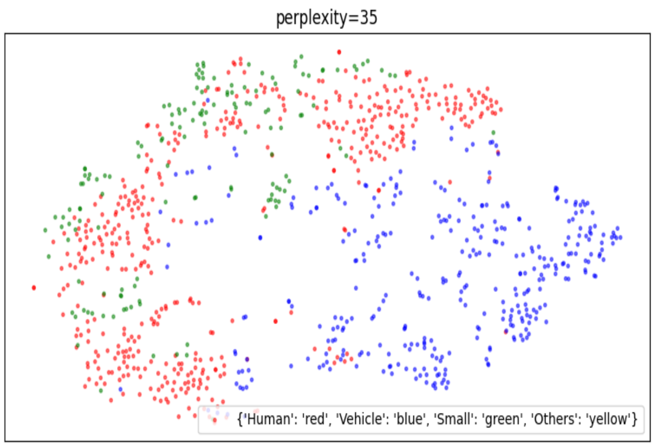

Figure 4.1: t-SNE plot of the embeddings output by the last hidden layer of a cross-entropy model on the train set.

4.4 Model Hyper-Parameters

The model which learns the class features, whatever kind it is of, is defined by a number of hyper-parameters related to the description given in Section 4.2 and are specified with lower and upper limits in Table 4.2. Note that some of the upper limits are kept lower than what may theoretically have been optimal in order to account for existing hardware constraints.

Table 4.2: Hyper-parameters with lower and upper limits specifying the models used for feature learning.

Hyper-Parameter	Lower	Upper
Kernel Size	2	3
Convolution Layers	1	2
Convolution Filters	1	8
RNN Units	8	64
RNN Layers	1	2

Chapter 5

Method and Results

This chapter focuses on the methodology employed to improve upon the baseline model and presents the primary results of this thesis. The first part describes how the contrastive loss function works as opposed to cross-entropy and outlines the approach taken to finding a good encoder network to be used as a backbone for other methods to build upon.

A number of models for classification are fitted to the embeddings output by the identified encoder. These methods relate to the concepts of random forest, logistic regression, K-nearest neighbors and principal components described in Ch. 2. At first, 4 experiments are carried out on the embeddings using the TPOT AutoML framework described in Section 4.1. A new encoder is then discovered for which PCA is combined with K-nearest neighbors in 4 new experiments. Additionally, a ”naive circle” model is defined for the sake of comparison.

The obtained results are presented and the latter part of the chapter studies the possibilities of binary classification and changing some features such as the window and data size.

5.1 Supervised Contrastive Loss

Recent work in the domain of self-supervised representation learning has resulted in new ideas related to the possibility of using a contrastive loss even in the case of fully supervised learning. In particular, this thesis draws inspiration from the work of [3] in which the authors leverage said advancements in self-supervised contrastive learning to develop the contrastive loss framework to the fully supervised case and substantiate some of its potential advantages over cross-entropy loss.

The version of contrastive loss used in this thesis is defined as follows. For two feature vectors x_1, x_2 with corresponding labels y_1, y_2:

$$L_{\text{con}} = (1 - Y)D^2 + Y \max(0, m - D)^2 \tag{5.1}$$

where $Y = 0$ if $y_1 = y_2$, where $Y = 1$ if $y_1 \neq y_2$ and where D is the distance between two embedded feature vectors generated by a model corresponding to x_1 and x_2 respectively. The parameter m is referred to as the margin and is a hyper-parameter which can be tuned on. It determines the distance which is required in embedding space before further separation no longer leads to continued improvement of the loss.

The idea of supervised contrastive loss is fairly straightforward: given a labeled data set and a suitable deep learning model, run gradient descent training of the model over the data set just like with cross-entropy but instead using the loss of Eq. (5.1) without a final classification layer. In doing so, the parameters of the model will be adjusted as to push points of opposing classes apart till the distance is at least m all the while pulling points from the same class closer and closer together.

The loss computation is carried out on the embeddings output by a projector net which is layered on top of the base model. Such a projector net could in principle be any network but in this work we employ a simple neural layer with 128 units. Once the training process is complete, the projector net is discarded and therefore has no direct relevance during inference.

Figure 5.1 contains a flowchart describing one forward pass during training with the base model architecture of this thesis. The input goes through the base model consisting of a CNN and an RNN and the subsequent output is normalized to the unit sphere in the same dimension as the number of values output by the RNN. It is then fed into a projector net consisting of 128 units whereby it is again normalized to the unit sphere and Eq. (5.1) is employed to compute the contrastive loss over a batch of outputs.

Once training is complete, any model which can handle the given number of classes and output embedded feature vectors can in principle be fit to those embeddings and produce classifications. The authors of [3] use a simple neural network after the encoder has been trained, however in this work we experiment with and develop alternative approaches for handling the embeddings.

Examples of models other than neural networks which we try fitting to the encoded embeddings are classification trees, random forests and logistic regression as well as a set of models based on principal component analysis and K-nearest neighbor classifiers.

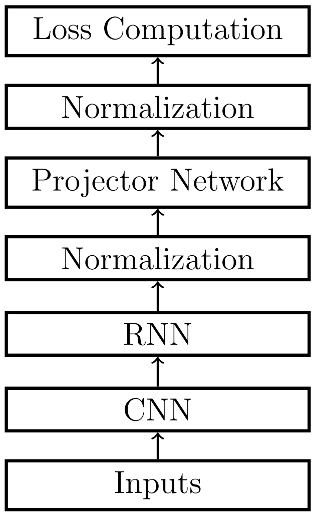

Figure 5.1: Flowchart of a single forward pass during supervised contrastive training. The feature inputs go through a the base model consisting of a CNN and an RNN and are then normalized to the unit hyper-sphere. These embeddings are then fed into a projector network consisting of 128 units and the contrastive loss is finally computed over a batch of such normalized outputs.

5.2 Training and Evaluation Protocol

Since the goal is to attain generalization to tracks not part of the training set and since frames within each track exhibit significant correlation, frames from the same track need to be used for only one purpose. That is, they are used either for training or validation but not both. Therefore, the following scheme was employed to allow for cross-validation:

(1) Tracks are separated into training and validation sets with a ratio of 4:1.

(2) Hyper-parameters determining the model and data format are determined.

(3) Windowing and augmentation of the data in the training set is carried out.

(4) The specified model is fitted to the training data and the validation score using a simple neural net classifier is recorded.

(5) The process is repeated until five validation scores have been calculated and the mean value is then computed.

After each pipeline has been assessed, the resultant validation score is associated to the hyper-parameters defining the pipeline and is used for eventually selecting the best set of such parameters. The chosen model is then fitted to the entire training data set.

To evaluate the model or models obtained in the training phase, a number of frames are selected from each track in the test set and the corresponding accuracy, precision, recall and F-scores are computed.

The test set in this context is not to be understood as a final assessment of a single selected model, as the selection procedure varies for the different methods utilized in this work, so that the scores from the training phase cannot be unambiguously interpreted, but rather as a mean to obtain such scores. These scores can then be used for final selection between models that were not obtained from the same selection protocol.

5.3 Finding an Encoder Network

With supervised contrastive loss and the selection procedure described in the previous section, the hyper-parameters of the model architecture outlined in Section 4.2 need to be discovered for a suitable encoder network to be specified. Once suitable hyper-parameters have been discovered, various changes either to individual such parameters or to the training procedure it self can be taken to assess the effect of the change. For this purpose, the following steps were taken in order:

(1) A set of 50 hyper-parameters were randomly sampled from a search space uniformly and the corresponding 5 fold cross-validation accuracies were computed on the train data set. The best model was selected.

(2) The best model from (1) was taken and refitted with larger batch size.

(3) The best model from (1) was taken and refitted with another RCS channel and with larger batch size.

(4) A hybrid model based on the one found in (2) incorporating frame-wise summary statistics was constructed.

In each case, the embeddings of the models were computed on the training set and the TPOT Auto-ML tool (see Section 4.1) was employed to find a suitable classifier. The models found by TPOT were all random forest classifiers except for Model 4 where a logistic regression model was selected.

Table 5.1 contains obtained recall and precision scores for the above four models whereas Table 5.2 contains accuracy and mean F-scores for the classes, also including the score of the baseline model for comparison. Out of the four models, Model 4 above incorporating frame-wise summary statistics has the highest scores overall followed by Model 2 without the summary statistics.

We chose Model 2 specifically as encoder network for further experimentation as it performs the best among the models without summary statistics while being simpler to work with than the one which includes these statistics.

Figure 5.2 displays t-SNE plots based on Model 2 both on the train and test set. It reveals the models apparent ability to separate the classes into clusters.

Table 5.1: Recall and precision scores for models (1)-(4) described in section 5.3.

Model 1	Human	Vehicle	Small
Recall	0.85	0.85	0.69
Precision	0.73	0.94	0.76

Model 2	Human	Vehicle	Small
Recall	0.85	0.88	0.75
Precision	0.79	0.95	0.75

Model 3	Human	Vehicle	Small
Recall	0.76	0.87	0.80
Precision	0.85	0.87	0.57

Model 4	Human	Vehicle	Small
Recall	0.88	0.92	0.71
Precision	0.81	0.96	0.80

Table 5.2: Accuracy and average F-scores for models (1)-(4) described in section 5.3 as well as those of the baseline model.

Score/Model	Baseline	1	2	3	4
Accuracy	0.85	0.82	0.85	0.81	0.87
Mean F-score	0.78	0.80	0.83	0.78	0.85

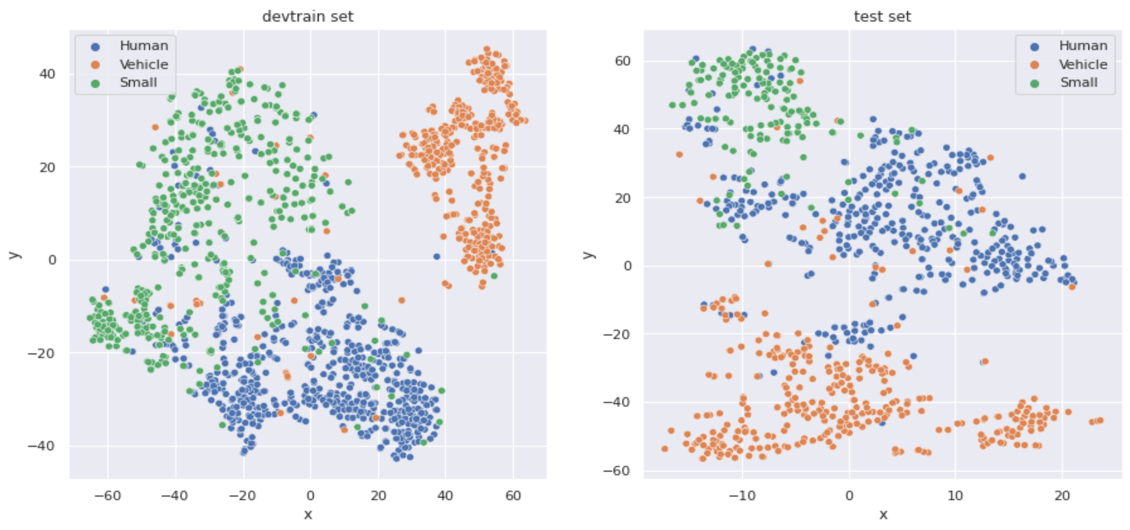

Figure 5.2: t-SNE plot of the observed embeddings on the first 21 frames of each track in the train and test sets of an encoder network fitted with batch size 4096.

5.4 Principal Component and Neighbor Methods

Instead of fitting a classifier directly to the embeddings output by an encoder it is possible to first project the embeddings to a lower dimensional space using a number of the principal components. The point is to reduce the prevalence of noise in the embeddings by preserving only some of the variance.

As a baseline model for reference in the context of this section, we consider "naive circles". For this model, circles with manually identified center-points and radii are drawn around the concentrations of human and vehicle points in the train set. Anything falling within those circles in the test set are said to belong to the corresponding class whereas anything falling outside is classified as Small. The reason for this choice is that it can serve to increase the precision scores for the Human and Vehicle classes, reducing the likelihood of false positives. The accuracy for this approach is 0.71. Recall and precision scores for this model along with center points and radii are presented in Table 5.3.

Figures 5.3-5.4 contain plots of PCA projected embeddings of Model 2 of Section 5.3. The first of the two shows how the embeddings look in 2 dimensions when a separate PCA matrix is fitted to either of the train and test set whereas the latter figure shows what the projected embeddings look like when using the PCA matrix from the embeddings on the train set. The main reason for producing these plots is to investigate the extent to which the PCA projection method is able to generalize whereby the embeddings of Figure 5.4 are to be compared to those in the right plot of Figure 5.3.

Table 5.3: Recall and Precision score for the naive circle model along with the circle center points and radii.

	Human	Vehicle	Small
Center	(0, 0.4)	(0.8, -0.2)	-
Radius	0.3	0.3	-
Recall	0.57	0.78	0.88
Precision	0.88	0.96	0.38

As an alternative to the parametric models of Section 5.3, we consider models based on K-nearest neighbors. These models are fitted either on the encoded features of supervised contrastive loss model directly or on their respective PCA-projected representations using one to four principal components. The same encoder and projection matrix is then used for inference on the test set. In either case, Model 2 from Section 5.3 is used as encoder network.

The scores of these models are presented tables 5.4-5.5 for 1-4 principal components and 150 neighbors which was identified to be a suitable number of points to include with the given data set.

Table 5.4: Recall and precision scores for the PCA and KNN based models. The variable n denotes the number of involved principal components and k the number of neighbors used.

n=1, k=150	Human	Vehicle	Small
Recall	0.83	0.72	0.90
Precision	0.79	0.81	0.91

n=2, k=150	Human	Vehicle	Small
Recall	0.86	0.92	0.84
Precision	0.87	0.91	0.84

n=3, k=150	Human	Vehicle	Small
Recall	0.86	0.92	0.90
Precision	0.90	0.90	0.85

n=4, k=150	Human	Vehicle	Small
Recall	0.87	0.70	0.92
Precision	0.78	0.89	0.91

Contrasting the naive circle approach with the PCA and KNN based methods we see that the accuracy is significantly lower for the naive circle than for each of the other methods. On the other hand, it does very well in terms of precision score for both of the Human and Vehicle classes with a Vehicle precision score of 96%. The only PCA-KNN model which performs better than the naive circle in terms of Human precision

Table 5.5: Accuracy and mean F-scores for models based on PCA and KNN.

Score/ Principal Components	1	2	3	4
Accuracy	0.84	0.88	0.89	0.85
Mean F-score	0.82	0.87	0.89	0.84

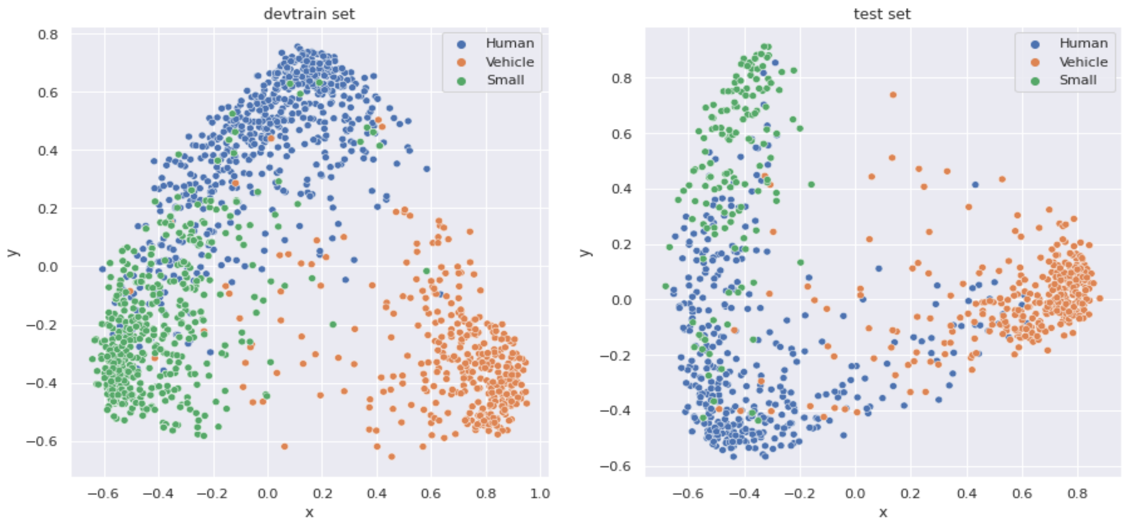

Figure 5.3: PCA plot of the observed embeddings on the first 21 frames of each track in the train and test sets of an encoder network. Note that there is a different projection matrix in each case, each of which being obtained from applying the encoder to the corresponding set.

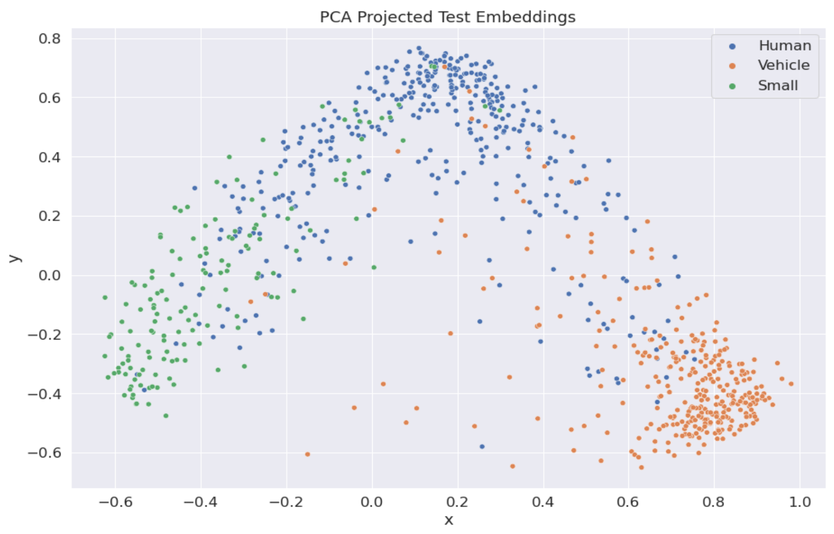

Figure 5.4: Plot of projected embeddings of the test set using the PCA projection matrix of the train set.

score is the one employing 3 principal components indicating that if the goal is to reduce Human false positives, this model is preferable considering its greater overall performance.

5.5 Binary Classification

In order to gain more control over how each classification is made we may cast the problem to be one of binary classification. By this we mean using the embeddings output by the encoder as before, but instead of fitting a final classifier to those embeddings which does ternary classification, we fit at least two separate classifiers which are defined in relation to a specific class such as Human or Vehicle.

This approach is mostly a change of perspective and is useful for convenience when it comes to studying various properties of the model. It is in contrast to starting over and refitting the encoder with different labels originally, which is ideally would require a new round of hyper-parameter selection and would be very time-consuming therefore.

If the goal is to reduce false positives of certain classes, such as Human or Vehicle, one can consider different thresholds for each of them and study how various scores such as the prediction accuracy and precision depend on the chosen threshold.

Figure 5.5 contains two plots that illustrate these relationships. In the left plot an object being classified as Human represents a positive outcome while any other class is considered negative. In the right plot this role is instead assumed by the Vehicle class. The models here employed where both KNN models fitted to embeddings output by the classifier that where projected onto three dimensional space using principal components.

In both cases, and especially in the Human case, there are clear relationships between the chosen thresholds and the scores. As the required threshold increases, the precision scores increase while the recall scores drop sharply. The accuracy on the other hand follows a parabolic pattern indicating that it serves as a kind of weighted measure of precision and recall.

Figure 5.6 contains corresponding ROC-curves for both the Human and Vehicle classes. The area under curve (AUC) is very high in the case of Vehicle, confirming its relative separability versus the other two classes. The AUC of Human on the other hand is likely suppressed by its overlap with the Small class.

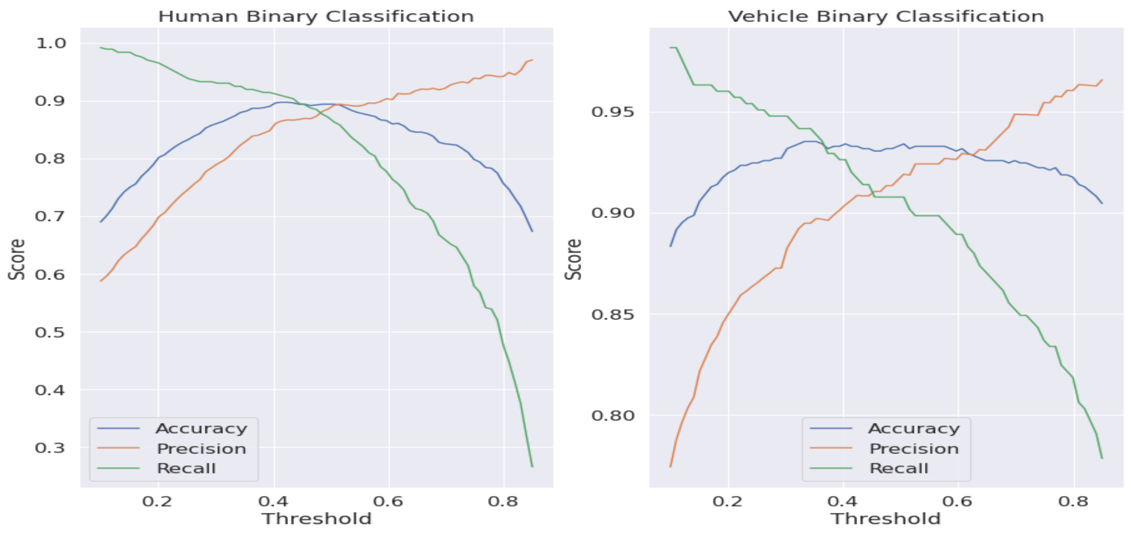

Figure 5.5: Plots illustrating the relationship between classification threshold and various scores of interest when the problem is cast to be one of binary classification where the model employed is based on PCA and KNN. To the left a positive represents a human classification and to the right a positive represents a vehicle. As the classification threshold increases, the precision score continues to improve at the expense of accuracy and recall.

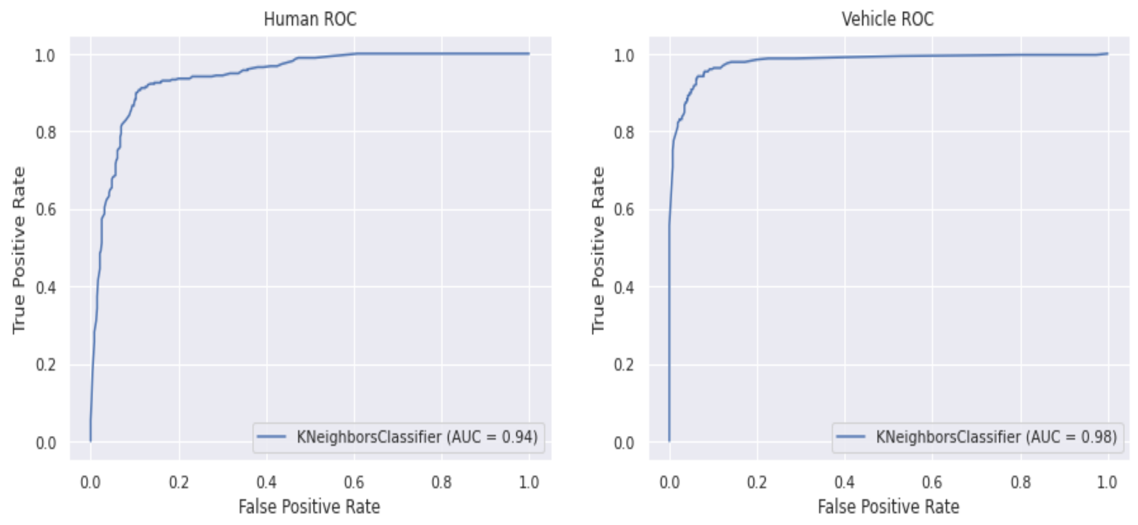

Figure 5.6: Receiver operating characteristic curves for human and vehicle binary classification respectively.

5.6 Assessing the Probability Distribution

By studying the distribution of predicted probabilities we can gain an understanding of how changing the classification threshold will affect the model behaviour, with the goal in mind to reduce the likelihood of false positives.

Figure 5.7 contains plots of the distributions of the maximum probabilities in two cases both based on supervised contrastive learning using ternary classification. Again, the models where fitted to the embeddings output by an encoder network projected to 3D space. The right plot show the probabilities of a cross-entropy neural net with exactly 3 neurons.

Somewhat surprisingly, the cross-entropy loss neural network in the right side of the figure has a smoother distribution which exhibits greater variability as measured by the standard deviation. This indicates that it may be preferable to simply use a dense layer as classification layer and then adjust the threshold accordingly.

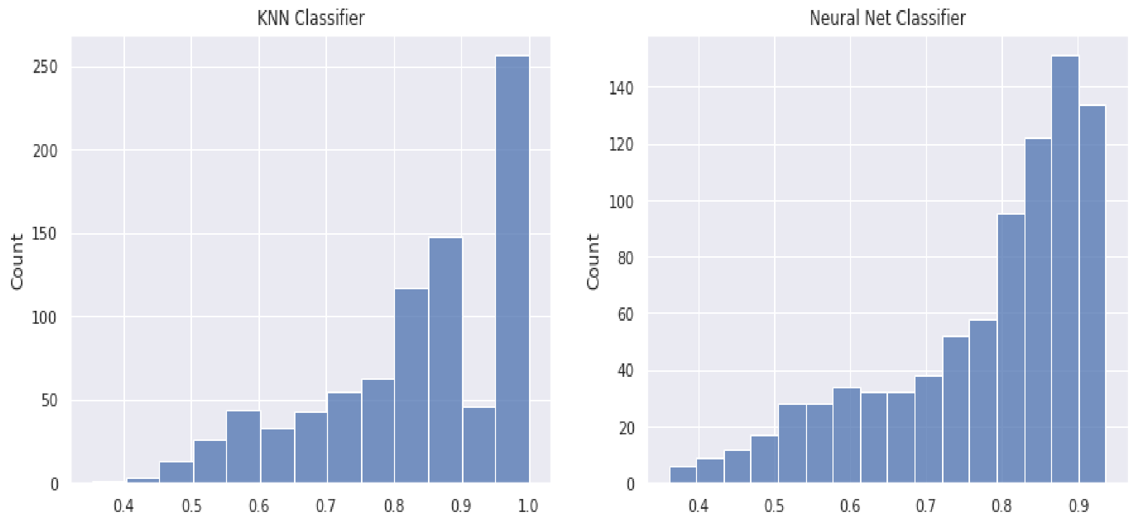

Figure 5.7: Distribution of the output maximum probabilities for two models fitted to the embeddings output from a contrastive loss based encoder. The embeddings where first projected down to three dimensions using principal components.

5.7 Varying the Window Length

It is not obvious which window and segment lengths are ultimately optimal given that windowing and segmentation is required at all. When using only one window however for classification, one may expect the accuracy score to increase as the length of the window increases, since each window then encodes more information. The downside is that as the number of tracks which are long enough to be included becomes smaller which may degrade performance. Therefore this section investigates the effect on accuracy depending on window length and segment size.

Figure 5.8 presents results for segment widths 5 and 7 where an analogous method to the one employed in Section 5.4 was employed with three principal components and where the number of neighbors was tuned on to maximize the accuracy. In the case of a window size of 35, segments of length 7 were employed. The score of the model with window size 21 is not the same as the best model presented in Section 5.4 since the batch size used in this case is lower.

The linear regression fit reveals a positive relation between the window size and accuracy with a coefficient of 0.003. Note that irrespective of the window size, the tracks tested on are the same, as only track with at least 35 frames are included for testing. This gives support to the idea mentioned above that longer windows yield better results.

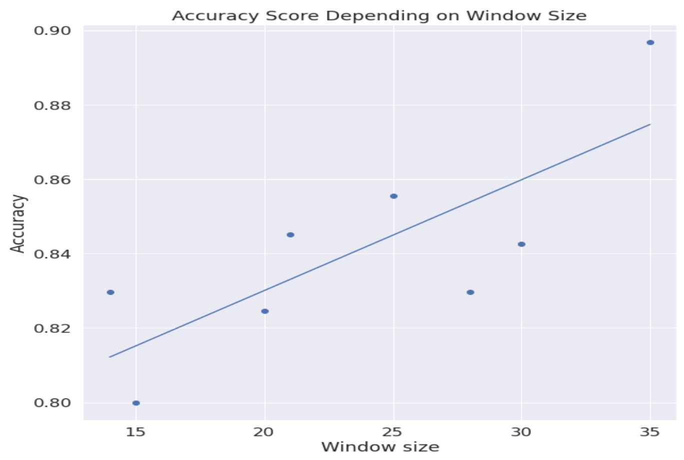

Figure 5.8: Accuracy scores depending on window length and segment width. In each case, a unique encoder is fitted with supervised contrastive loss with the given window size and segment width. The employed batch size is 512. Then, a PCA matrix with 3 principal components is fitted to the embeddings where after KNN is used to fit a classifier. Lastly, the performance is evaluated on the test set.

5.8 Varying the Data Size

In order to assess the quality of the data and judge the need for more we fit the best performing model to a variable proportion of the train data and evaluate the result on the test set. Figure 5.9 contains a plot of the relationship between the train data proportion and the accuracy score. As the amount of data increases, so does the score. Even though there seems to be diminishing returns, which is to be expected, the graph indicates that there is room for further improvement by gathering more high quality data for the models to be fit on.

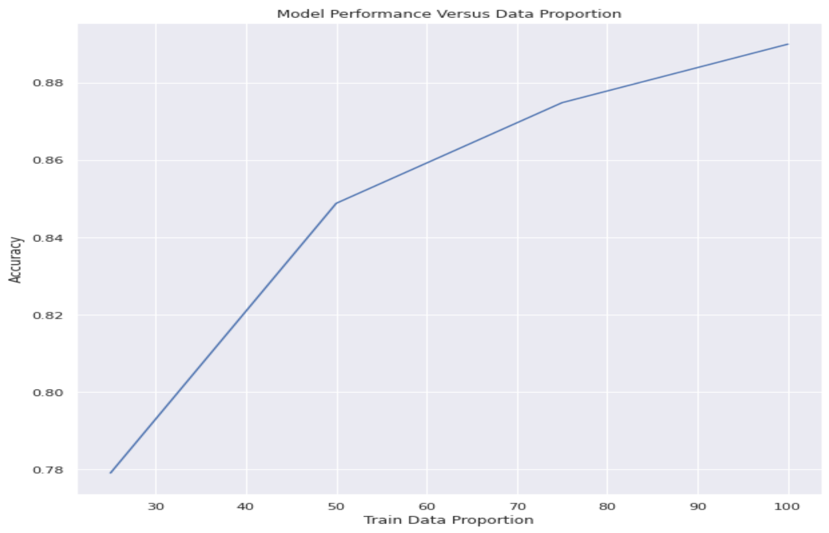

Figure 5.9: Accuracy scores depending on the proportion of the train data included when fitting the best performing model.

Chapter 6

Discussion

Here we discuss some of the results and methods of Chapter 5 with a broader view as well as possible paths for future work that could yield additional improvements over those attained in this thesis.

6.1 The Problem of Under-Specification

The role of the test set, in terms of how it has been used in this work, has grantedly been more like a validation set in relation to how one would customarily think of a test set. This was a conscious decision and the idea was to use it instead as a means to test new ideas and methods as opposed to holding it away and waiting for the final model to be assessed only once. Additionally, there is the potential problem of under-specification (see [10]) which poses the possibility that even given a good final score, the model may end up behaving in very unexpected ways when tested in various context-contingent applications. As the data we use is not necessarily taken from the same distribution as the final use-case of the system will entail, considering that the radar system can be set up differently and in wildly different spaces, a single number from a final test set is not likely to be particularly definitive anyway.

6.2 Uncertainty Estimation and Thresholding

Once different methods have been developed, new solutions can be layered on top to add functionality and the final test of the model or method will be how it performs in practise.

One such possibility for added functionality is uncertainty estimation. In the case of an encoder fitted with supervised contrastive loss and PCA, a KNN model implicitly encodes uncertainty estimation in terms of which classes the considered neighbors belong to. If there are say 100 neighbors considered and 95 of them belong to the Human class, then this gives a lot more confidence in the final classification than if only 55 of them belong to Human and say 45 belong to Small. Therefore, if the goal is to reduce the number of false positives in terms of say Human classifications, one could impose a differential threshold on the required class proportions among the neighboring points.

Figure 5.5 reveals the trade-off between three main scores of interest in the case of such differential thresholds. It is apparent that the greater the required threshold the lower the likelihood of false positives both for the Human and Vehicle classes. It is also apparent that the recall score along with the accuracy score to some extent begin to drop drastically as the threshold approaches 1, which indicates that it is necessary to consider the choice of threshold with some caution in order not to over-emphasize the importance of precision over recall and accuracy. At around a threshold of 0.6 in the case of Human it is worth noting however that the accuracy of this model is in fact superior to that of the baseline model with scores presented in Table 4.1 while the precision score is about 0.9 to be compared with 0.83 in the cross-entropy case. This seems like a major improvement if the goal is to reduce false positives in the application and depending on the use-case one could imagine a scenario where the threshold is adjusted as to optimally fit the need of the situation.

The AUC scores of Figure 5.6 confirm that the Vehicle class is the most distinct class in the data set as the value is very close to one. For Human the optimal ratio between true positive rate and false positive rate seems to be about 9:1.

Interestingly, Figure 5.7 reveals that a simple single layer cross-entropy based neural network may be equally well suited for such uncertainty classification as KNN. This neural network had only three neurons and was fitted to the projected training embeddings output by the encoder in just three dimensions. Therefore the likelihood of over-fitting is low and the network simply learns to delineate the regions corresponding to the various classes in such a way that points near the edges are given probability estimates closer to uniform. This is in stark contrast to how cross-entropy normally behaves when fitted to the input features directly, where the output embeddings may be highly non-linear and susceptible to over-fitting.

It is conceivable that the problem of under-specification is in fact improved by the PCA projection discussed above, or alternatively could be so by instead fitting a two, three or four dimensional dense layer as the final layer in the encoder network directly.

It could be argued that the main principal components have the effect of preserving only the variance that counts and that much of the noise is reduced by the projection, with the consequence that the model will tend to behave in a more predictable manner even in potentially different circumstances, which would be desirable and conducive to reproducibility. This is because by the very nature of the training procedure with supervised contrastive loss, the most distinguishing features of the classes will contribute the most the variability in the set of encoded features, which is preserved by the principal components.

6.3 Suggested Application Scheme

One could take Model 3 of Table 5.5 which was based on the second encoder of Section 5.3 and do classification repeatedly with some suitable stride. With each classification, a different threshold for the classification could be chosen in such a way that the required probability for a positive classification declines with each iteration. This would be motivated by the observation that Human tracks in particular tend to be the longest in the data set and one would always chose the last classification as the one with priority. This could be done by employing binary classification in two steps. At first, the model would test whether or not the observed track corresponds to a vehicle and since the vehicle class is well separated, a positive classification here could be treated as definite. If the model determines the target track is not a vehicle, it would be made to repeatedly check whether it is human or not for as long as possible or for a maximum amount of time with a time-dependent threshold. In any case this needs to be investigated to determine its potential.

6.4 Model Contrastive Power

The idea behind refitting the best identified model of Section 5.3 with larger batch sizes was that it may serve to improve the contrastive power of the model, both in terms of allowing for closer mapping of points of the same classes and in terms of greater separation between classes in embedding space. Comparing the scores presented in Table 5.2 between Model 1 and Model 2 this seems to have been successful to some extent

The amount of models fitted when doing hyper-parameter selection in Section 5.3 was 50 which is a relatively small number. The reason for this is that it takes a great deal of time to do 5-fold cross validation on any one particular set of hyper-parameters with the given data set and contrastive loss even on a modern GPU. Therefore it is unlikely that the resultant hyper-parameters are the absolutely best ones out there. Looking at figures 5.2-5.4 it becomes clear however that the identified parameters generates a model which is capable of learning the features of the involved classes in a way which generalizes and the scores presented in tables 5.1-5.2 validate the hypothesis that cross-entropy can be outperformed by alternative methods based on contrastive loss.

Comparing Figure 4.1 with the contrastive loss embeddings, we see that the separation between Human and Small classes is especially unclear in the case of cross-entropy. This may have the effect during inference that small corruptions or noise to the input features perturb the embeddings just enough to enter a region where classifications are incorrect. In the supervised contrastive loss case however this effect may be improved by the greater separation in the embeddings to begin.

6.5 Future Work

Since the radar point-cloud data was unstructured in terms of the number of points per frame, the intermediary representation of frame-wise histograms was created to allow for a CNN to process. It may be possible to find a model which can anyhow process the point-clouds directly or one could try to employ an alternative representation of some sort. Appendix A presents and describes one such plausible alternative. With this representation, some experiments with simple neural nets and a CNN-RNN model yield accuracy scores in the range of 75-80% which could likely be improved by doing hyper-parameter selection on the level of the model selection and training process as well as with the parameters regulating the representation itself.

There is the possibility of investigating the effect of using a different version of the contrastive loss function such as the ones presented and analyzed in [3]. This was not done in this thesis as the implementation is not trivial and as we did not know of any ready to use implementation of these presented loss functions as of its writing. Since the supervised loss function of Eq. (5.1) is quite simple in nature, it is possible that a more sophisticated version could have yielded superior results.

Even though deep learning models have the ability to learn effective predictors and output probabilities for each prediction, they lack the ability to assess the certainty of the probabilities that they output. That is to say, just because a probability output by the model is close to 1, does not mean that absolute confidence in its prediction is warranted if the input features it had been trained on differ greatly from what is being observed. One interesting and recent idea in the field for how to account for this limitation is to use so-called deep evidential learning. In [11], the authors use the theory of subjective logic to develop a framework for uncertainty estimation. They do so by placing a Dirichlet distribution on the class probabilities as to treat the class probabilities as subjective opinions and a function that collects evidence is learnt. This is an interesting avenue which could be attempted also for the models of this thesis.

A common approach for making the most out of the available data is to use bootstrapping to train N models and use all of them at inference by averaging the predictions. The only obvious downside to this approach is that storing all the models and having them all make predictions can come to be expensive. Given powerful hardware this may not be much of a problem however when this is not the case it may be intractable. It may however be possible to distil the knowledge of an ensemble of models into a few models or a single model that can be used for inference [12].

Bibliography

[1] Sainbayar Sukhbaatar et al. "Training convolutional networks with noisy labels". In: *arXiv:1406.2080* (2014).

[2] Zhilu Zhang and Mert Sabuncu. "Generalized cross entropy loss for training deep neural net-works with noisy labels". In: *Advances in neural information processing systems* (2018).

[3] Prannay Khosla et al. "Supervised Contrastive Learning". In: *arXiv:2004.11362* (2020).

[4] *Radar*. 2021. URL: `https://en.wikipedia.org/wiki/Radar`.

[5] *Radar Cross-Section*. 2021. URL: `https://en.wikipedia.org/wiki/Radar_cross-section`.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[7] Randal S. Olson and Jason H. Moore. "Stochastic Neighbor Embedding". In: *Neural Information Processing Systems* (2002). `https://cs.nyu.edu/~roweis/papers/sne_final.pdf`.

[8] Sandeep Rao. *Introduction to mmwave Sensing: FMCW Radars*. 2021. URL: `https://training.ti.com/sites/default/files/docs/mmwaveSensing-FMCW-offlineviewing_2.pdf`.

[9] Randal S. Olson and Jason H. Moore. "JMLR: Workshop and Conference Proceedings". In: *International Conference on Machine Learning* (2016). `http://proceedings.mlr.press/v64/olson_tpot_2016.pdf`, pp. 66–74.

[10] Alexander D'Amour et al. "Underspecification Presents Challenges for Credibility in Modern Machine Learning". In: *arXiv:2011.03395* (2020). `http://proceedings.mlr.press/v64/olson_tpot_2016.pdf`.

[11] Murat Sensoy, Melih Kandemir, and Lance M. Kaplan. “Evidential Deep Learning to Quantify Classification Uncertainty”. In: *CoRR* abs/1806.01768 (2018). arXiv: `1806.01768`. URL: `http://arxiv.org/abs/1806.01768`.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: `1503.02531 [stat.ML]`.

Appendix A

Alternative Point-cloud Representation

In spite of working relatively well, the frame-wise histograms which have served as the main point-cloud representation in this thesis suffer from a number of drawbacks. For one, spatial structure is lost as values such as RCS and $\dot{r}$ are disassociated from each other and from their corresponding x and y coordinates.

Also, even if temporal patterns can be observed in the representation, there is no straightforward way of employing a sequential processing model such as an RNN directly. This was solved by instead using a CNN, which even though it has the ability to capture temporal, is a method for processing spatial data.

This problem arises because each frame, which is a natural unit for time in the context of point-clouds, is essentially reduced to a single dimension due to the histogram binning procedure.

As alternative representation of radar point-clouds is illustrated in Figure A.1. For this representation, each frame is instead treated as an image with 2 color channels corresponding to RCS and $\dot{r}$ respectively. Each frame is centered by subtracting the (x, y) center-point from the (x, y) coordinates of each point in the frame. The area around the origin is discretized into a 2D grid of pixels where the RCS and $\dot{r}$ values are binned respectively resulting in a data structure that can be visualized as a sequence of images. In the figure, the frames progress from the upper left corner to the bottom right corner.

The resultant point-cloud representation may be referred to as a radar video since each frame in this representation acts just like a frame in video generated by camera. This allows for a CNN to process each frame like an image, just like for camera, and the output from the CNN can be passed on to an RNN for sequential processing.

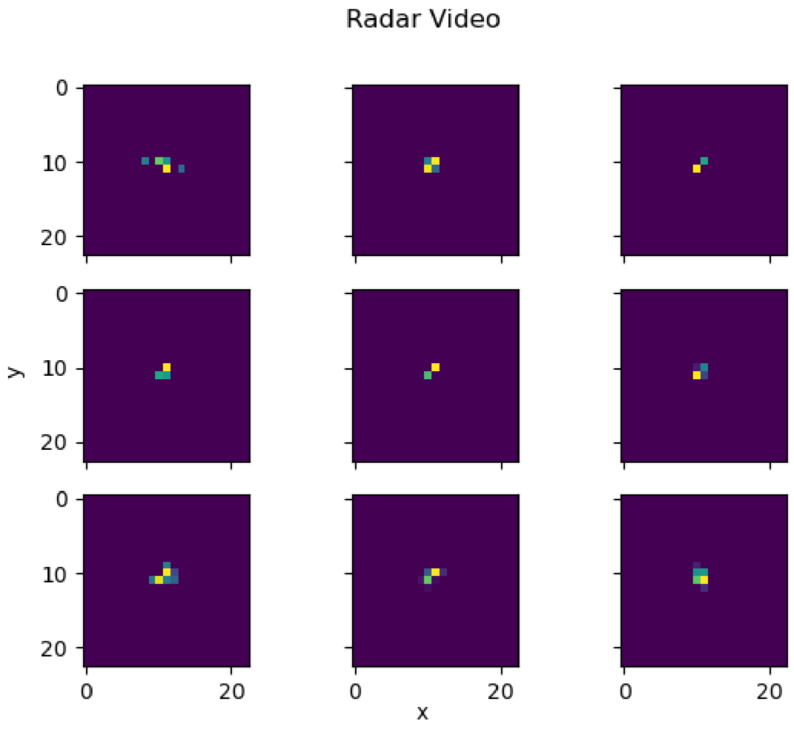

Figure A.1: Example of track being visualized as a radar video. Here each image corresponds to a singe frame. Each frame is centered by the mean (x, y) coordinates and the point RCS and $\dot{r}$ values binned into specified pixels in the area around the origin of the plane. There are two channels whereof the RCS channel is seen in the image. The frames progress from upper left corner to bottom right.