CHALMERS
UNIVERSITY OF TECHNOLOGY

On Classification of Road Types for Automotive Applications

Master's thesis in Complex Adaptive Systems

JEANETTE WARNBORG

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis EX005/2018

On Classification of Road Types for Automotive Applications

JEANETTE WARNBORG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2018

On Classification of Road Types for Automotive Applications
JEANETTE WARNBORG

Supervisor: Kenny Karlsson, Aptiv
Examiner: Jonas Fredriksson, Department of Electrical Engineering

Master's Thesis EX005/2018
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image captured from one of Aptiv's test vehicles.

Gothenburg, Sweden 2018

On Classification of Road Types for Automotive Applications
JEANETTE WARNBORG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

One of the challenges within autonomous driving is for the vehicle to determine what kind of surrounding environment it is operating in. This information could assist the vehicle in its decision making. In this project two methods, based on neural networks and support vector machines, for determining road types using radar and image data have been compared. The road types were divided in to three classes, highway, major road and city. The best result was achieved by a neural network using image data. Using radar data gave the worst results and that had a negative effect on the classification using radar and image data in combination.

Keywords: Autonomous driving, Neural networks, Support vector machines

Acknowledgements

I would like to thank Kenny Karlsson for his support throughout all parts of the project. Furthermore I would also like to thank the examiner Jonas Fredriksson for his help and feedback.

Jeanette Warnborg, Gothenburg, 2018

Contents

List of Figures

List of Tables

1
Introduction

In recent years the area of autonomous driving have gained a lot of attention in both the public and scientific communities. For a car to be able to drive itself safely and efficiently, it need to be able to make at least as good decisions as a human driver. This problem is not solved trivially and one crucial part of this problem is how the car should determine in what kind of environment it is driving. The aim of this thesis is to further investigate how such a classification system could be implemented.

1.1 Background

The idea of fully autonomous vehicles are getting more real rapidly. This means that the vehicles has to make smart and safe decisions.

Aptiv's Active Safety department is developing sensors used by vehicle manufacturers worldwide. The sensor data is used as input to numerous advanced active safety functions such as Adaptive Cruise Control, Automatic Emergency Braking, Queue Assist and are also being used in the development of autonomous driving systems. An autonomous driving system needs to be able to identify and adapt to its surroundings, e.g. drive slowly when pedestrians are nearby or when driving on bad roads. To enable the car to make these decisions it need to have enough information about its surrounding environment. A study comparing how well different classification methods perform in different environments and in determining whether the car is driving on- or off-road showed that it is possible to classify the surrounding with a high level of accuracy (Tang and Breckon, 2011). As input they use images from which they extract regions of interest. For this thesis the input will not only contain images but also radar tracking data.

1.2 Contributions

The main contributions of this project are the comparison of two different classification methods (neural networks and support vector machines) and three different data types (radar, images and radar and images combined). Also included is an in depth discussion regarding future work and how to further improve the results.

1.3 Aim

This master thesis will serve to investigate how various sensors, such as radars and cameras, can be used to assess the situation around the vehicle. More specifically we will classify sensor data from a car in to one of three different environments: **Highway** have at least two driving lanes separated with a central barrier, with no sharp turns, high speed and normally no pedestrians nearby. **Major road** is similar to the highway but can have a single lane, has a slightly lower speed, more turns and not necessarily a central barrier. **City** environments can look quite different with a mixture of pedestrians, cyclists, cars and public transportation vehicles. Samples from these three classes are presented in Figure 1.1. The classification will be done in order to be able to determine e.g. maximum appropriate speed in real-time depending on what type of road the vehicle is being driven. Further on this could be used to limit the maximum speed for a vehicle in a specific environment. A high level sketch of the classification pipeline is shown in Figure 1.2.

(a) Sample image from the highway data set.

(b) Sample image from the major data set.

(c) Sample image from the city data set.

Figure 1.1: Sample images from the different environments.

Figure 1.2: High level sketch of the classification pipeline. Radar data, images and the two combined will serve as input. The classification method is either a neural network or a support vector machine and the output is the assigned class of the corresponding input.

1.4 Limitations

In this thesis we will only consider three previously mentioned environments. The data used to train the algorithms will only represent these three situations. We also assume favorable conditions, both to simplify the gathering of the data but also the classification. This corresponds to data gathered during day time in nice weather without rainfall.

1.5 Specification of issue under investigation

In this project the main focus will be on the following questions:

- Using tracker data from radars together with images from the front camera, is it possible to determine if the car is driving on a highway, major road or in an urban environment?
- With what accuracy can the road type be classified?
- Which classification methods gives the highest accuracy?

1.6 Outline

This thesis is organized as follows: In chapter 2 the theory behind neural networks and support vector machines is presented together with a short section on image preprocessing. The setup of the data collection and the data used for this project is described in chapter 3. In chapter 4 the classification algorithms that were evaluated are described in detail. The results that were found for the different methods and data types is presented in chapter 5. Finally, in chapters 6 and 7 a discussion of the results and methods is found together with the conclusion.

2
Supervised learning

In this chapter we introduce the techniques and concepts used in this project. More specifically we show how to classify different road types using both neural networks and support vector machines. Firstly, in sections 2.1, 2.1.1 and 2.1.2 we give a basic introduction to neural networks and in sections 2.1.3 and 2.1.4 we describe the two different types of neural networks that will be used for this project. In section 2.1.5 we give a short introduction to TensorFlow, the framework used to implement the neural networks. Secondly, in section 2.2 we give a basic introduction to support vector machines and in sections 2.2.1 and 2.2.2 we describe the kernels, types and extensions that will be used for this project. In section 2.3 we describe a method to extract features from images to be used as input to the support sector machine.

2.1 Neural network (NN)

A neural network could be described as a computational graph where each node performs some kind of computation. A node in such a graph is often referred to as a layer and each layer consists of one or many neurons. In Figure 2.1 a visualization of a simple network can be seen. Typically the graphs are directed and acyclic. Neural networks can be used for both unsupervised and supervised learning and we will from now on limit this introduction to explain supervised learning only. This requires labelled data for the training, which often needs to be manually labeled. The layers can have different layouts and be advantageous for different tasks. Also by combing different types of layers it is possible to build networks that are optimized for different purposes. This introduction will, however, focus on the architecture suitable for classification of radar and vision data, both separately and combined. The output from a node in one layer multiplied with the specific weight connecting that node to a node in the next layer added to a bias will serve as input to the node in the next layer:

$$I_{i+1} = f(I_i w_i + b_i) \ ,$$

where I_{i+1} is the output from layer i and the input to layer $i+1$. w_i is the weight matrix for layer i, b_i is the biases for layer i and f is the activation function.

In the case of classification the output from the final layer is often the class affiliation probability distribution. This is compared to the actual labels and the error is computed, often with the squared error

$$E_{\text{total}} = \frac{1}{2} \sum (\text{target} - \text{output})^2 \ . \tag{2.1}$$

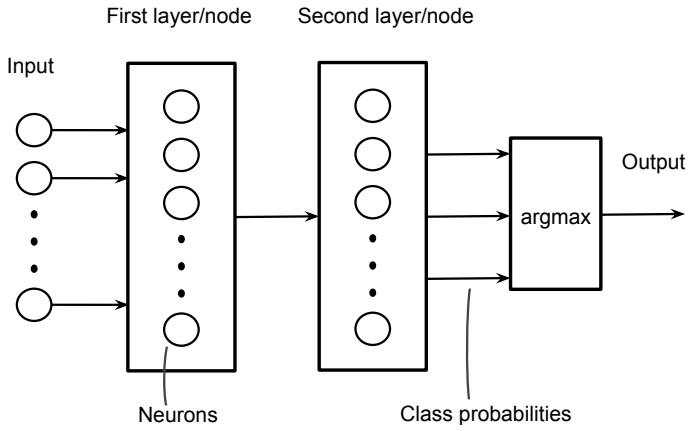


Figure 2.1: A simple description of a neural network with two layers. The data is fed through the layers, producing a probability vector of the different classes from which the output class can be found by taking the argmax of the output vector.

Since we want the network to produce the same output as the target, (2.1) can be viewed as the objective function to a minimization problem. To accomplish this an algorithm that iteratively updates the weights and biases, also known as back-propagation, described in section 2.1.2, is most often used. A more comprehensive introduction to neural networks is found in Shanmuganathan and Samarasinghe (2016).

2.1.1 Activation function

The activation functions of a neural network serve two purposes. In the output layer the activation function is used to produce an output in the same form as the target. For the other layers the activation function corresponds to a nonlinear transform of the data that is passed between the layers. One of the most common activation functions is the sigmoid function

$$h(x) = \frac{1}{1 + \mathrm{e}^{-x}}$$

with the codomain equal to the interval $(0, 1)$. Another common activation function that often is used in the output layer of a classification network is the softmax function. It will normalize a vector $\mathbf{x}$ with arbitrary (real) values to a vector σ with values between 0 and 1 and unit length,

$$\sigma_j(\mathbf{x}) = \frac{e^{\mathbf{x}_j}}{\sum_{i=1}^{K} e^{x_i}} \ , \quad j = 1, \ldots, K \ ,$$

where K is the number of classes. Finally the rectified linear unit activation function (ReLu)

$$R(x) = \max(0, x)$$

has increased in popularity in recent years. Results have shown that the time needed to train networks is reduced significantly when using this function in favor of e.g. sigmoid or tanh function (Krizhevsky et al., 2012).

input: target t and data x
init weights w and biases b **repeat**
  prediction $p \leftarrow$ forwardpropagation(x)
  for *all layers in reverse order* **do**
    compute error $\delta(p, t)$
    compute $\Delta w(\delta)$
    compute $\Delta b(\delta)$
    $w \leftarrow w + \Delta w$
    $b \leftarrow b + \Delta b$
  end
until *stopping criteria is reached*;
Algorithm 1: Pseudo code for a backpropagation algorithm. How to compute Δw and Δb depends on which optimization algorithm used.

2.1.2 Optimization

By minimizing (2.1) the performance of the network will improve. One way of doing this is by using gradient based methods, such as stochastic gradient decent or Adam (Kingma and Ba, 2014), applying them from the output of the network and backwards. By doing this iteratively while passing new data through the network, it will gradually adapt the weights and biases to the data. The algorithm is commonly know as Backpropagation and is described in Algorithm 1.

One of the crucial steps in the algorithm is to compute the gradients between the layers. If the absolute value of the input to the sigmoid activation function is large, the gradient will go towards zero. When this happens the weights and biases of those neurons of the network will not be updated which can eventually lead to 'dead' neurons in the network. This is why it is important to initialize the weights and biases carefully and also to normalize the input data so that the input to the activation function does not grow too large.

It is common to pass the entire training set through the backpropagation algorithm many times and each pass of the complete training set is called an epoch. Often, the time it takes to train a network is defined by the number of epochs needed to train the network.

2.1.3 Multilayer perceptron (MLP)

In a MLP all layers are fully connected, that is all neurons in the previous layer act as input to all neurons in the next layer. This is how layers in regular neural networks are connected. A MLP consists of at least three layers, an input layer, one or many hidden layers and an output layer. The hidden layers are used for nonlinear fit to the data. In Figure 2.2 we can see an example of the architecture of a MLP.

2.1.4 Convolutional neural network (CNN)

A convolutional neural network is used for tasks such as image classification. The convolutional layer of a network is based on the assumption that a layer can form a general impression of the input even though a single neuron is only allowed to

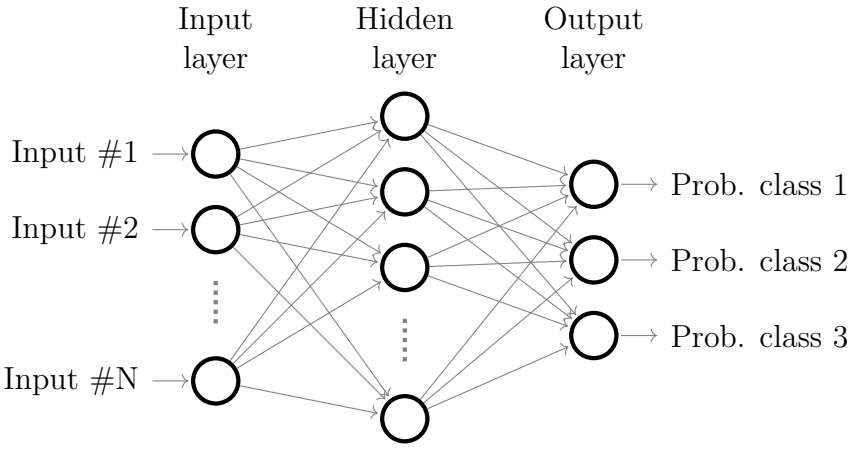


Figure 2.2: Example of structure for a simple MLP.

observe a small part of the input rather than the entire input. A convolutional layer consists of multiple filters that realize the assumption above. Eventually these filters will learn to detect different defining features of the input, such as edges in an image (Ciresan et al., 2012).

2.1.4.1 Pooling layer

After a convolutional layer a pooling layer can be added to downsample the size of the output. It serves two purposes, to reduce the computational complexity and to control overfitting. This is done by applying a aggregation function on a moving window over the data and combining the results. For images, it is common to use pooling layers with a window size of 2×2 that will output the maximum value from each window. In addition to the window size and aggregation function there is one more parameter, the stride, which controls the number of steps the window is moved in each direction.

2.1.5 Tensorflow

The framework to implement neural networks used for this project is called TensorFlow (Abadi et al., 2015). It is a framework developed by Google which is a very powerful and configurable framework written in C++ and Python. It supports distributed and GPU accelerated training for efficient training on large datasets.

2.2 Support Vector Machine (SVM)

A SVM is a binary classification algorithm. The algorithm uses supervised learning to create a hyperplane that separates the data in the two classes with the largest possible margin. The standard implementation used today was first proposed by Cortes and Vapnik (1995).

Given a linearly separable dataset:

$$(\bar{x}_1, y_1), (\bar{x}_2, y_2), \ldots, (\bar{x}_n, y_n)$$

where $\bar{x}_i$ is a p-dimensional data point and $y_i \in \{-1, 1\}$ indicates its corresponding class, the SVM algorithm finds a $(p-1)$-dimensional hyperplane that separates the two classes. This plane is characterized by the parameters b and $\bar{w}$ and the classifier can be defined as

$$y_{\text{new}} = \text{sgn}(\bar{w} \cdot \bar{x}_{\text{new}} - b) \ .$$

To achieve a robust classifier, the margin is maximized by solving the optimization problem

$$\begin{aligned} &\underset{\bar{w}}{\text{minimize}} \quad ||\bar{w}|| \\ &\text{subject to} \quad y_i(\bar{w} \cdot \bar{x}_i - b) \geq 1, \ i = 1, \ldots, n \ . \end{aligned} \tag{2.2}$$

The resulting hyperplane is completely determined by the $\bar{x}_i$ nearest to it which are also called support vectors. This formulation of the SVM is also know as a hard margin SVM. An example of a hard margin SVM is illustrated in Figure 2.3a, in which the data is linearly separable.

For data that is not linearly separable the optimization problem to minimize is instead

$$\left[\frac{1}{n}\sum_{i=1}^{n} \underbrace{\max(0, 1 - y_i(\bar{w} \cdot \bar{x}_i - b))}_{\substack{=0 \text{ if } \bar{x}_i \text{ lies on the correct} \\ \text{side of the margin}}}\right] + \lambda||\bar{w}||^2$$

where λ is a regularization parameter that determines the trade off between the size of the margin and ensuring that $\bar{x}_i$ lies on the correct side of the margin (Wu and Liu, 2007; Cortes and Vapnik, 1995). The parameter λ is often found by the use of cross validation. This formulation of the SVM is know as a soft margin SVM. An example of a soft margin SVM is illustrated in Figure 2.3b, in which the data is linearly separable with some data points that is located on the wrong side of the margin.

2.2.1 Kernels

For data that is not linearly separable and clustered in a way that makes separation by a straight line meaningless the kernel-trick can be used. The idea is to have a function that maps the feature vectors to a high dimensional space in which the data is linearly separable. If this function is non-linear the resulting classifier will also be non-linear in the original feature space (Theodoridis and Koutroumbas, 2008). More specifically, the kernel function

$$K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$$

is a similarity measure between two feature vectors and defined as the dot product of the high dimensional mapping functions, φ:

$$K(\bar{x}_i, \bar{x}_j) = \varphi(\bar{x}_i) \cdot \varphi(\bar{x}_j) \ .$$

A classifier can then be defined as

$$y_{\text{new}} = \text{sgn}\left(\left[\sum_{i=1}^{l} c_i y_i K(\bar{x}_i, \bar{x}_{\text{new}})\right] - b\right)$$

where c_i and b are coefficients found by optimization and l is the number of support vectors. Note that only the support vectors are needed to define the classifier.

Some of the most common kernels are polynomial and radial basis functions (RBF). The polynomial kernel is defined as

$$K(\bar{x}_i, \bar{x}_j) = \left(\bar{x}_i^\top \bar{x}_j + c\right)^d$$

where c and d are parameters of the kernel. The RBF kernel is defined as

$$K(\bar{x}_i, \bar{x}_j) = \exp\left(-\frac{||\bar{x}_i - \bar{x}_j||^2}{2\sigma^2}\right)$$

where σ is a parameter of the RBF kernel. An example of a SVM with a RBF kernel can be seen in Figure 2.3c.

2.2.2 Implementations and extensions

One of the most common types of the SVM algorithm is the C-SVM (Schölkopf et al., 2000). It has a parameter C that controls the regularization. Another common type is the ν-SVM that very similar to C-SVM, with the only difference that $C \in [0, \infty)$ and $\nu \in [0, 1]$. Both of these are implemented in the LIBSVM library (Chang and Lin, 2011).

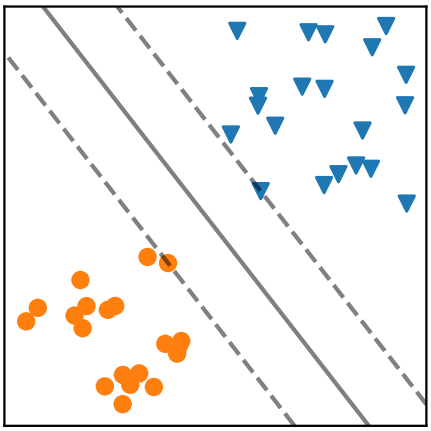
A common extension is the Multi-class SVM (MSVM) that allows for classification of more than two classes. Two of the approaches are one-versus-one classification and one-versus-all classification (Duan and Keerthi, 2005; Hsu and Lin, 2002). In one-versus-one classification a single classifier for each pair of classes is trained and the data points are evaluated in every classifier. The labels for these data points are then decided by a majority vote by the classifiers. In one-versus-all classification one classifier for each class is trained. A classifier for class i is trained with class i as positive label and all other classes with negative label.

2.3 Histogram of Oriented Gradients (HOG)

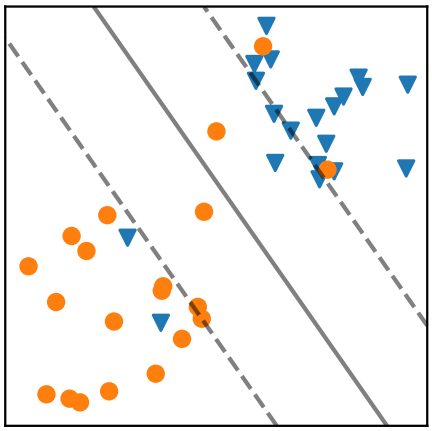
One way to accentuate important features in an image and at the same time reducing the size of it is to calculated the HOG of the image (Dalal and Triggs, 2005). This method is used on gray-scale images and done as follows. First, the images is divided in patches. For each pixel in a patch the gradient vector $\bar{x}_i = (x_{1,i}, x_{2,i})$, is defined as

$$\begin{pmatrix} x_{1,i} \\ x_{2,i} \end{pmatrix} = \begin{pmatrix} x_{\text{left},i} - x_{\text{right},i} \\ x_{\text{above},i} - x_{\text{below},i} \end{pmatrix}$$

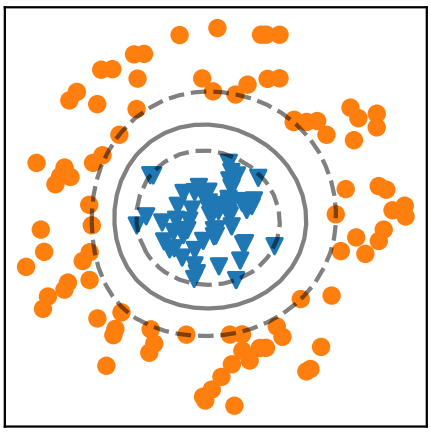
where $x_{\text{left},i}$, $x_{\text{right},i}$, $x_{\text{above},i}$ and $x_{\text{below},i}$ corresponds to the values of the pixels to the left, right, above and below pixel i. Secondly, the magnitudes of the gradients is aggregated in a histogram over the gradient directions. An illustration of this procedure is given in Figure 2.4. Finally, the histograms from all patches are concatenated into a single feature vector that can be used as input to a classification algorithm, such as a SVM.



(a) Hard margin SVM with linearly separable data. All data points are classified correctly.



(b) Soft margin SVM that allows for wrongfully classified data points in order to maximize the margin.



(c) Linearly inseparable data classified using RBF kernel.

Figure 2.3: Three cases of classification using support vector machines. Blue triangles and orange dots represent samples from two classes. The solid line is the decision boundary that separates the classes. The distance between the solid line and the dotted lines is the margin. The dotted lines are parallel to the solid line and intersect the support vector(s).

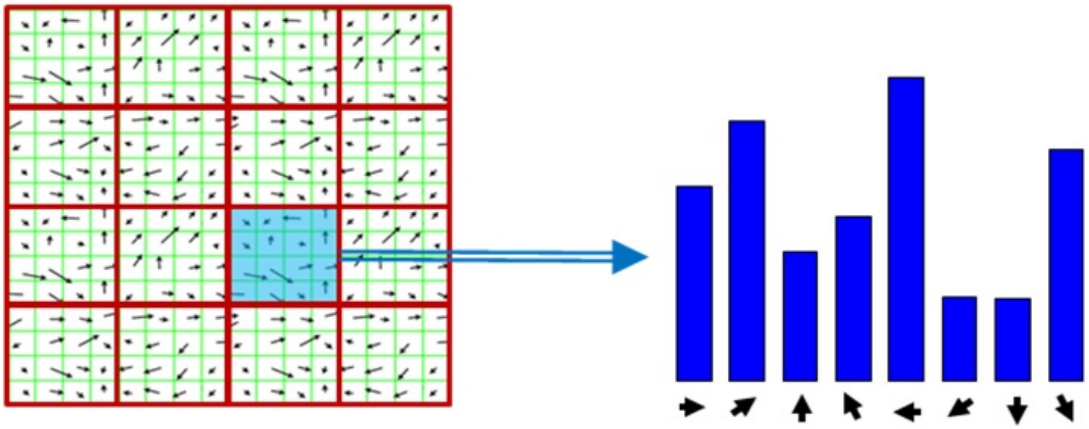


Figure 2.4: To the left: A picture divided in patches reduced to the gradient vector of each pixel. To the right: The histogram of oriented gradients for a single patch. Image courtesy of Gil (2013).

3
Data

In this chapter a detailed description of the data used in this project is presented. The method for gathering the data is introduced in section 3.1, this is followed by more detailed descriptions of the method in sections 3.2 and 3.3. In sections 3.4 and 3.5 the radar and image data preprocessing is outlined.

3.1 Data gathering

The data for this project was gathered using a car equipped with a front camera and a front radar. A route containing the three environments (highway, major road and city) selected for classification was chosen. The data was gathered on a single day and is evenly distributed between the environments. The data from these logs where used to construct the training, validation and test data sets.

3.2 Setup

The data were collected using a car equipped with a RACam forward-looking camera, capable of detecting vehicles, pedestrians, bicyclists, road edges, lane markings and traffic signs and a RACam radar, which is an electronically scanned 76 GHz automotive radar mounted behind the windshield.

A typical video log is about one minute long and has approximately 1448 frames, this means that a lot of the frames look really similar. From these frames every 50th frame is extracted and used for the training and validation with the corresponding radar data. The frames from the video files were extracted as gray scale images of size 640×480. The result was images in the same format as the sample images in Figure 1.1.

The radar data used in this project is not raw unprocessed signals from the radar. With software from Aptiv the detections from the radar are grouped together and tracked as possible objects. These objects are given statuses and real valued measurements for different properties and is stored in a struct. Four of these measurements are listed in Table 3.1.

3.3 Scope

Since all data was gathered at the same time during a single day, the outer conditions, like weather and time of day, is relatively invariant across the data set. This is

Table 3.1: Table of tracklet measurements.

Name	Description
`radar_cross_section`	A measure how detectable an object is by radar
`vcs_long_posn`	Longitudinal position of object w.r.t. host
`vcs_lat_posn`	Latitudinal position of object w.r.t. host
`vcs_long_vel`	Longitudinal velocity of object w.r.t. host

important to make sure that the classifier is not biased towards e.g. certain rain, sun or daylight conditions. From the generated data, some of the radar data and the video logs were extracted.

3.4 Radar data preprocessing

There are 64 slots for objects in both long and mid range which means that a maximum of 128 objects can be stored and used. Not all of these properties contains relevant information for this project. First, four different properties are selected and extracted from the initial struct. For each frame in the video log the mid and long range data is found and stacked together, resulting in a feature vector with length 128. Each feature was normalized to zero mean and unit variance.

3.5 Extracting features from vision data for SVM

Since SVMs does not work well with large amounts of data it is important to extract important features from the images for the training. This was done by computing the HOG for each image. Additionally the images were randomly skewed to reduce the negative effects of any unwanted rotation of the camera.

4
Algorithms

In this chapter the algorithms used for classifying the different road types are described. First, a description of the shared configuration between the neural networks is given in section 4.1. This is followed by more detailed descriptions of the neural networks specific configurations are presented in sections 4.1.1, 4.1.2 and 4.1.3. Finally, the setup for the MSVM is presented in section 4.2.

4.1 Neural network

Three NNs were implemented. For classification of the radar data a MLP was used. For the vision and combined data two different CNNs were implemented. For the convolutional and fully connected layers the weights were initialized as random numbers drawn from a truncated normal distribution with mean zero and standard deviation 0.05. In the truncated normal distribution values are re-sampled if their magnitude is more than 2 standard deviations from the mean. The biases were initialized to a constant value close to zero, 0.05. In all networks the Adam optimizer was used with learning rate $\lambda = 10^{-4}$ and other parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$ as defined by Kingma and Ba (2014).

The data for the different networks was divided in three sets, a training, a validation and a test set. The size of the training set contained 3/5 of the complete set and the test and validation sets contained 1/5 each. The validation set was used during training to find the best network parameters and avoid overfitting. For each epoch the validation set was evaluated and when a new minimum value for the validation loss was found the current network was saved.

The test set was used after the training was done to evaluate the accuracy of the network on never before seen data.

Because of the randomized initialization of the network parameters each network was trained and evaluated using different seeds to the random number generators. In total, for each data type four separate networks were trained and evaluated. This was done to be able to give a more robust estimate of the accuracy.

4.1.1 Image classifier

For the image classification a CNN was implemented. The input consisted of images as described in section 3.2, but resized to 128×128.

The network structure consists of three convolutional layers. The first two layers had 32 filters of size 3×3. The third layer had 64 filters of size 3×3. Each

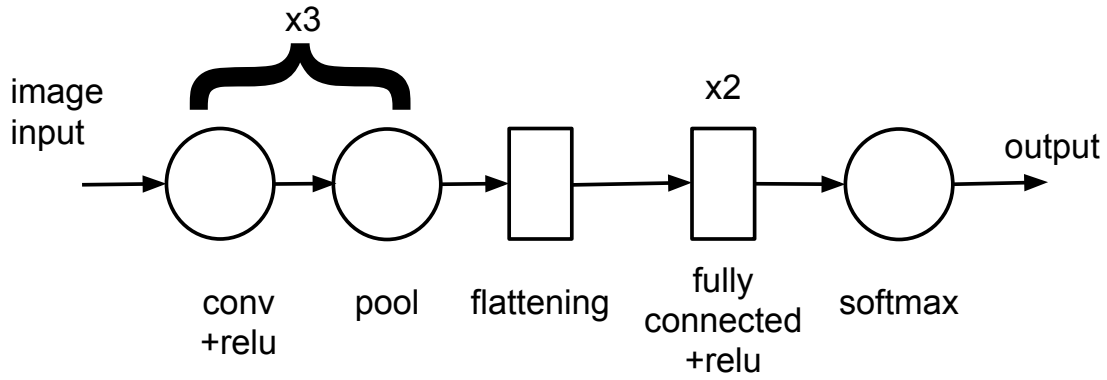


Figure 4.1: A schematic view of the image classification network.

convolutional layer was followed by a max pooling layer and a ReLu activation function. The pooling layer had a window size of 2×2 and a sliding window stride of 2×2. The convolutional layers were followed by a flattening layer that reshapes the data to a one dimensional vector and two fully connected layers with 256 neurons in each layer. Finally, to produce an output with the probability distribution over the classes the softmax function was used. The network is visualized in Figure 4.1.

4.1.2 Radar classifier

To classify the radar data a simple MLP was used. Out of the available signals from Table 3.1, `radar_cross_section` and `vcs_long_posn` were selected as input to the network. We choose to use only two signals for computational simplicity. Empirical tests showed that these two signals gave acceptable results. The network structure

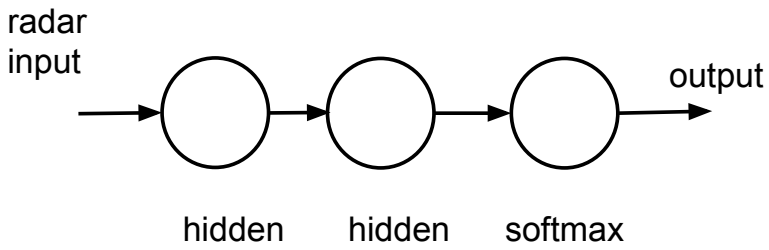


Figure 4.2: A schematic view of the radar classification network.

consisted of an input layer for the 256 features, two hidden layers with 200 and 256 nodes respectively and an output layer for the class probabilities. The activation function used for the hidden layers was the sigmoid function. The activation function used for the output layer was the softmax function. The network is visualized in Figure 4.2.

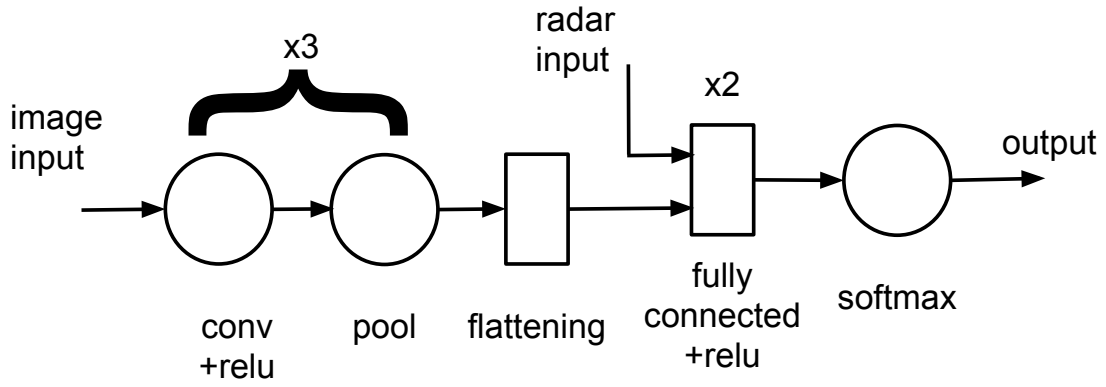


Figure 4.3: A schematic view of the combined classification network.

4.1.3 Combined network

The first part of the combined network is the same as the first part of the image classifier. Three convolutional layers with max pooling and ReLu activation functions followed by a flattening layer. The input to these layers consisted only of the images. The output from the flattening layer was stacked together with the same radar data that were used in the radar classifier. This served as input to the next, fully connected layer. This was followed by another fully connected layer with three outputs that were fed through a softmax function to produce the probability distribution. The network is visualized in Figure 4.3.

4.2 Support vector machine

Three one-versus-one MSVM classifiers were implemented, one for classifying image data, one for radar data and one for the combined data. The classifiers were built using the scikit-learn library (Pedregosa et al., 2011).

In order to get an unbiased result, the entire data set was split in to a test and training set. The test set was used to evaluate the classifiers. The training set was used to first find the best value of c and then used for training of the classifiers. The selection of c was done using 10-fold cross validation as follows: For each $c \in \{10^{-7}, 10^{-6}, \ldots, 10^{7}\}$ one classifier was trained and evaluated for each fold. The c-value for which the classifiers had the highest average validation accuracy over the 10 folds was chosen for the final classifier.

The radar data for the MSVM consisted of the same signals as for the MLP, as described in section 4.1.2. The input to the image classifier consisted of their corresponding HOGs, as described in section 3.5. The input to the combined classifier consisted of the stacked of the input to the radar and image classifiers. Additionally the data was scaled before training and validation according to

$$x_i \leftarrow \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

where x_i corresponds the value of the ith feature, $x_{\min}$ and $x_{\max}$ corresponds to the

minimum and maximum values for the ith feature across the entire data set.

A RBF kernel was used for the image and combined classifiers and a polynomial for the radar classifier.

5
Results

In this chapter the results from the different classifiers and data sources are presented. The results for the NN and SVM classifiers are presented in the following sections.

In Table 5.1 a summary of the test set accuracy of the classifiers is presented. In total the best result is achieved using image data only with the CNN. We can also note that the accuracy using the radar data is low compared to the image data.

Table 5.1: Highest accuracy reached for different methods and inputs. Since the NNs were evaluated many times the mean and standard deviation is presented.

	Radar data	Image data	Combined data
NN	$49.0 \pm 1.2\%$	$96.9 \pm 1.0\%$	$94.8 \pm 1.2\%$
SVM	59.2%	95.2%	92.1%

5.1 NN – Image classifier

In Table 5.2 the results from the runs using the CNN with image data is presented. For the test data the average accuracy and standard deviation was 96.9 ± 1.0 % and for the entire data set 99.0 ± 0.4 %.

The confusion matrix for this classifier is presented in Figure 5.1a. It shows that the classifier does not have any significant bias towards any of the classes. The most common misclassification is major or highway classified as city.

Table 5.2: The accuracy for the runs using the CNN on the image data with the mean and standard deviation. For both the test set and the entire data.

	Test data	All data
Run 1	98.1%	99.4%
Run 2	97.3%	99.2%
Run 3	95.8%	98.6%
Run 4	96.3%	98.6%
Mean	96.9%	99.0%
Std	1.0%	0.4%

5.2 NN – Radar classifier

In Table 5.3 the results from the runs using the MLP with radar data is presented. For the test data the average accuracy and standard deviation was 49.0 ± 1.2 % and for the entire data set 56.2 ± 2.4 %. Compared to the other two networks the time per epoch was much shorter. The number of epochs needed to reach 100% training accuracy and to show signs of overfitting was also much lower for this network.

The confusion matrix for this classifier is presented in Figure 5.1b. It shows that the classifier is heavily biased towards the highway class and that it is unable to classify samples from the major class with any significant accuracy.

Table 5.3: The accuracy for the runs using the CNN on the radar data with the mean and standard deviation. For both the test set and the entire data.

	Test data	All data
Run 1	47.2%	52.9%
Run 2	49.7%	56.0%
Run 3	49.3%	57.4%
Run 4	49.7%	58.5%
Mean	49.0%	56.2%
Std	1.2%	2.4%

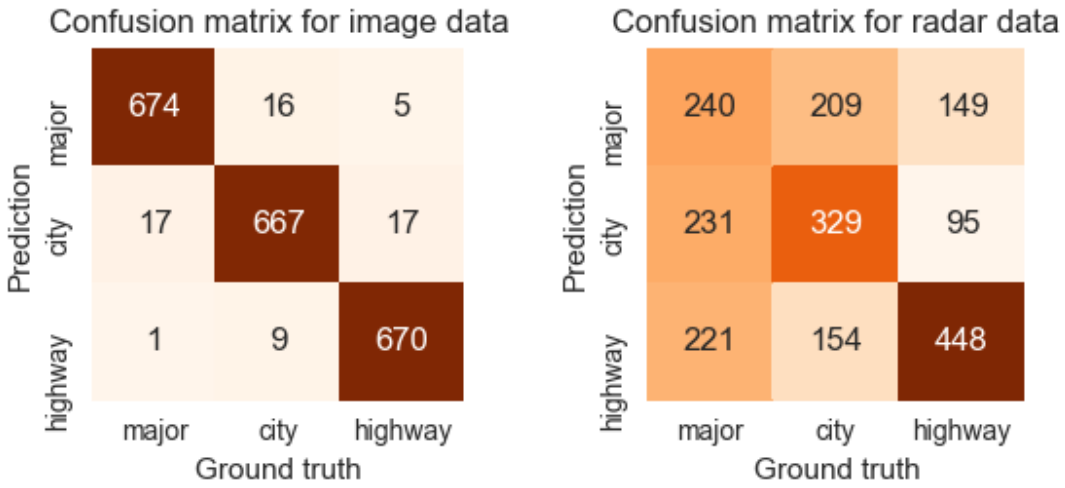
5.3 NN – Combined classifier

In Table 5.4 the results from the runs using the CNN with image data is presented. For the test data the average accuracy and standard deviation was 94.8 ± 1.2 % and for the entire data set 98.2 ± 0.5 %.

The confusion matrix for this classifier is presented in Figure 5.1c. Similar to the confusion matrix for the image data, this classifier does not have any significant bias towards any of the classes. The most common misclassification is major or highway classified as city.

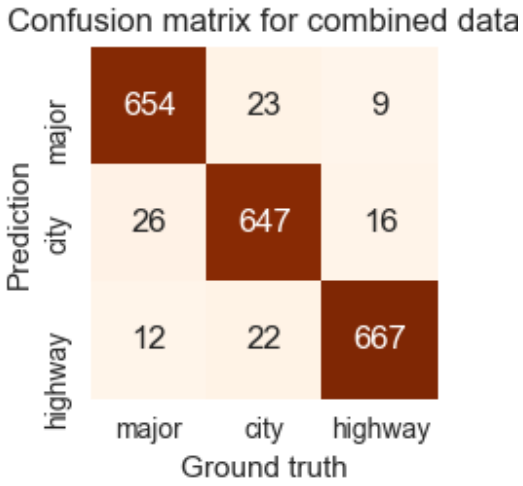
Table 5.4: The accuracy for the runs using the CNN on the combined data with the mean and standard deviation. For both the test set and the entire data.

	Test data	All data
Run 1	95.2%	98.5%
Run 2	96.1%	98.7%
Run 3	94.6%	98.1%
Run 4	93.3%	97.6%
Mean	94.8%	98.2%
Std	1.2%	0.5%



(a) Confusion matrix heatmap for the image data.

(b) Confusion matrix heatmap for the radar data.



(c) Confusion matrix heatmap for the combined data.

Figure 5.1: Combined confusion matrix heatmaps of all the runs for the different NN classifiers. In the confusion matrix the element at index i, j correspond to the number of samples from class j classified as class i. The confusion matrix for a perfect classifier would have 0 at the off diagonal elements. The sum of row i corresponds to the total amount of samples that was classified as class i and the sum of column j corresponds to the total number of samples of class j.

5.4 SVM – Image classifier

For the test data the classification accuracy was 95.2% with $c = 10^5$ and RBF-kernel. For the entire data set the accuracy was 99.1%.

The confusion matrix for this classifier is presented in Figure 5.2a. It shows that the classifier does not have any significant bias towards any of the classes.

5.5 SVM – Radar classifier

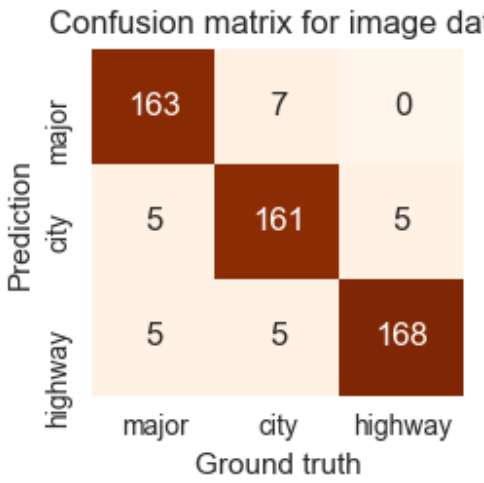
For the test data the classification accuracy was 59.2% with $c = 10^6$ and polynomial kernel with $d = 3$ and for the entire data set the accuracy was 92.2%.

The confusion matrix for this classifier is presented in Figure 5.2b. It shows that the classifier is heavily biased towards the major class.

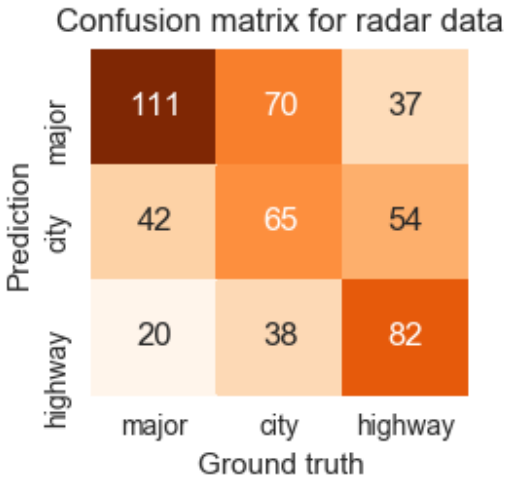
5.6 SVM – Combined classifier

For the test data the classification accuracy was 92.1% with $c = 100$ and RBF kernel and for the entire data set the accuracy was 96.4%.

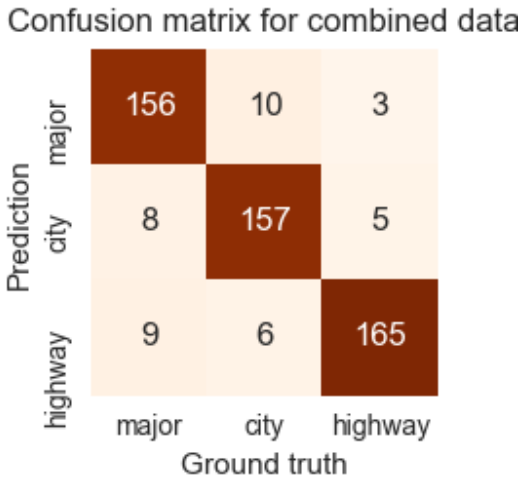
The confusion matrix for this classifier is presented in Figure 5.2c. It shows that the classifier does not have any significant bias towards any of the classes.



(a) Confusion matrix heatmap for the image data.



(b) Confusion matrix heatmap for the radar data.



(c) Confusion matrix heatmap for the combined data.

Figure 5.2: Confusion matrix heatmaps for the different SVM classifiers. In the confusion matrix the element at index i, j correspond to the number of samples from class j classified as class i. The confusion matrix for a perfect classifier would have 0 at the off diagonal elements. The sum of row i corresponds to the total amount of samples that was classified as class i and the sum of column j corresponds to the total number of samples of class j.

6
Discussion

In this chapter we analyze the results of chapter 5 and give suggestions to possible ways of further improving the results. The main focus of this chapter is in the analysis of the neural network classifiers in section 6.1. The results from the SVM classifiers are analyzed in section 6.2 and suggestions for future work are given in section 6.3.

6.1 Neural network

In Table 5.1 we see that the image classifier has the highest classification accuracy and that the radar classifier has significantly lower classification accuracy than the other two classifiers.

In Figure 5.1 we see the confusion matrices for the NN classifiers. For the major and city classes the combined classifier is most likely negatively affected by the radar data since the radar classifier had the most problems in classifying those classes.

In the following subsections we will discuss and analyze the performance of the different NN classifiers.

6.1.1 Image classifier

The image classifier had the best classification accuracy of all the classifiers. The average classification accuracy from Table 5.2 was $96.9 \pm 1.0\%$. This result can be compared to the commonly used CIFAR-10 benchmark dataset for image classification (Krizhevsky and Hinton, 2009) where the state of the art classification accuracy is 96.5% (Graham, 2014). The CIFAR-10 dataset consists of 60000 32×32 colour images divided in to ten classes, with 6000 images per class.

Even though these results are similar one has to consider the differences between the datasets and the task at hand. Since CIFAR-10 has ten classes, the classification task is more complex and difficult compared to the three classes in the dataset used for this thesis. Additionally, the difference between the images within the classes is larger in CIFAR-10 than in the images used for this thesis which makes the classification task more complex in the CIFAR-10 case.

On the other hand, the difference of the images between the classes should be greater in CIFAR-10, which contains classes of e.g. frogs and airplanes, compared to the images used in this thesis. The CIFAR-10 set also contains more samples, which is advantageous when training any network.

Given the comparison above, it should be possible to achieve a higher classification accuracy using additional data and further improvements to the model.

6.1.2 Radar classifier

The radar classifier had the worst classification accuracy of all the classifiers. The average classification accuracy from Table 5.3 was $49.0 \pm 1.2\%$. During training of the network it became clear that the network suffered badly from overfitting as the validation loss reached its minimum value after only a few epochs and thereafter increase.

Overfitting in a neural network can be tackled in a variety of ways. Adding more training data is always useful, as it will increase the variance of the training data, resulting in a more generalized model. It is also possible to implement some kind of regularization such as dropout (Srivastava et al., 2014).

In comparison to the SVM for radar classification there should be room for improvement of the MLP given the large difference in classification accuracy. This could be done by refining the architecture and parameters of the network.

Another way to improve the classification accuracy could be to include additional measurements from the radar data. Given the very low classification accuracy in the current state, it is likely that a combination of all of the suggestions above needs to be implemented to reach a satisfactory classification accuracy.

6.1.3 Combined classifier

The average classification accuracy for the combined classifier was $94.8 \pm 1.2\%$, as seen in Table 5.4. The initial idea behind the combined classifier was that the features extracted from the image data combined with the radar data would be easier to classify than to classify them separately. This does not seem to be the case however, as the image classifier has a higher classification accuracy. Instead, it seems as though the radar data impairs the performance of the network. This should be expected given the very low classification accuracy of the radar classifier.

To improve the combined classifier the suggestions mentioned in sections 6.1.1 and 6.1.2 should both be implemented. Most important should be to increase the size and variation of the data sets and also to investigate which measurements to extract from the radar data.

6.2 MSVM

In Table 5.1 we see that the image classifier has the highest classification accuracy of the MSVM classifiers and that the radar classifier has significantly lower classification accuracy than the other two classifiers.

In Figure 5.2 we see the confusion matrices for the SVM classifiers. The overall bad performance of the radar classifier probably affects the combined classifier negatively.

Except for the radar data classifier the SVM based classifiers performed worse than the NN classifiers. While there potentially is a lot of room for improvements

in the NN classifiers, it is not obviously the same for the SVM classifiers. It is likely that improving the image preprocessing and feature extraction and experiment with other combinations of the radar measurements would give the most gain. It is also possible to tune the parameters and choice of kernel to further improve the performance of the classifiers.

6.3 Future work

There are several different tracks to follow in order to achieve better results. Perhaps the most important track is to acquire more data. This data should be more varied within the classes to improve the training and verify the robustness of the classification methods. Since the radar data was the hardest to classify, the radar classifiers has probably the most room for improvement. This includes the choice of measurements which could for instance contain the number of identified pedestrians. If adding more measurements were to be done, then additional preprocessing of those measurements could be needed as well.

The CNN used to classify the image data in this project is comparatively simple against many of the state of the art architectures. Using techniques from these architectures the results should be able to be improved. The most important improvement for the neural networks in general is to implement regularization, such as dropout (Srivastava et al., 2014) to reduce the overfitting that occurs.

Finally, there is always possibilities of achieving better results by choosing the parameters of the classifiers more carefully. This applies for the neural networks as well as the support vector machines.

7
Conclusion

In this project two different methods for classifying road types have been evaluated against each other. The road types considered were highway, major road and city. This was done using radar and camera data collected from the area surrounding Gothenburg. The first method was based on convolutional neural networks and multi layer perceptrons and the second method was based on support vector machines. Each of the methods have been evaluated with radar and image data separately as well as the combination of the two.

The best result was achieved with a convolutional neural networks using image data only. It was closely followed by the support vector machine also only using image data as input. Both methods performed poorly using only the radar data which had a negative effect on the results when using the combined data.

To further improve the results more data should be gathered to ensure the robustness of the classifiers. It should be possible to improve the accuracy of the classifiers by further review of the architecture and parameters.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3): 1–27, 2011.

D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. URL `https://doi.org/10.1007/BF00994018`.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

Kai-Bo Duan and S. S. Keerthi. *Which Is the Best Multiclass SVM Method? An Empirical Study*, volume 3541, pages 278–285. Springer Berlin Heidelberg, 2005.

Levi Gil. A short introduction to descriptors, 2013. URL `https://gilscvblog.wordpress.com/2013/08/18/a-short-introduction-to-descriptors/`. Accessed: 2018-01-23.

Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014. URL `http://arxiv.org/abs/1412.6071`.

Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

Subana Shanmuganathan and Sandhya Samarasinghe. *Artificial Neural Network Modelling*, volume 628. Springer International Publishing, 1st 2016 edition, 2016.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

I. Tang and T. P. Breckon. Automatic road environment classification. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):476–484, 2011.

Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, 4th Edition*. Academic Press, 4 edition, 2008.

Yichao Wu and Yufeng Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.