UNIVERSITY OF AMSTERDAM

MSc ARTIFICIAL INTELLIGENCE

MASTER THESIS

Real-time detection of traffic behavior using traffic loops

by

THOMAS VAN DER HAM

10551905

June 28, 2018

Number of Credits: 36
Januari 2018 - Juli 2018

Supervisor:
prof. dr. ir. B.J.A. KRÖSE

Assessor:
dr. A. VISSER

COVALENT

Abstract

Traffic accidents and stopped vehicles in tunnels can result in dangerous situations. Fast responses to these kinds of events are essential to limit the danger and to restore the traffic flow. In most tunnels, inductive traffic loops are used in combination with a rule-based system to alert the traffic controller. The rule-based system that is currently used gives many alerts to the traffic controller which do not require his or her attention. Because of these false alarms, traffic controllers stop relying on the existing system. Such false alarms occur mostly during rush hour traffic.

The goal of this thesis is to better inform the traffic controllers by using machine learning to find different types of traffic behavior. By doing this the traffic controller can determine what to do based on the type of traffic. A Multilayer Perceptron was trained as a binary classifier to distinguish between normal and abnormal traffic. The existing rule-based system was used as a baseline for the binary classifier. In order for the MLP to outperform the rule-based system historical data was necessary during classification. Using only the current data of the traffic loops was not enough for the MLP to outperform the rule-based network.

The rule-based network scores an F_1score of 0.6601. While the MLP using only the current data scores an F_1score of 0.5249. However, if historical timesteps are used it scores an F_1score of 0.7872. The data generated by the traffic loops is a sequence of passing cars, so Recurrent Neural Networks and Hidden Markov Models were also used because they are designed for sequence data. The Recurrent Neural Networks were the best performing machine learning methods. Both a LSTM and a GRU were used and the GRU performed the best with an F_1score of 0.8236. While the Hidden Markov Models were not able to outperform the rule-based system with an F_1score of 0.0059.

Contents

1 Introduction **1**
- 1.1 Research questions . . . 2
- 1.2 Contributions . . . 3

2 Related work **5**
- 2.1 Inductive traffic loops . . . 5
  - 2.1.1 Speed approximation . . . 5
  - 2.1.2 Vehicle classification . . . 6
  - 2.1.3 Accident detection . . . 6
- 2.2 Sequence classification . . . 6
  - 2.2.1 Feature-based classification . . . 7
  - 2.2.2 Distance-based classification . . . 7
  - 2.2.3 Model-based classification . . . 8

3 Experimental setup **10**
- 3.1 Dataset . . . 10
- 3.2 Visualization . . . 11
- 3.3 Annotation . . . 12
  - 3.3.1 Data preprocessing . . . 13
- 3.4 Evaluation . . . 14
  - 3.4.1 Metrics . . . 14
  - 3.4.2 Test set . . . 15
  - 3.4.3 Cross-validation . . . 16

4 Method **17**
- 4.1 Multilayer Perceptron . . . 17
  - 4.1.1 Imbalanced data . . . 18
  - 4.1.2 Layouts . . . 19
- 4.2 Recurrent Neural Networks . . . 20
  - 4.2.1 Comparison with Multilayer Perceptrons . . . 22
  - 4.2.2 Layouts . . . 22
- 4.3 Hidden Markov Model . . . 23

5 Results **25**
- 5.1 Baseline . . . 25

5.1.1 A priori probability 25
5.1.2 Rule-based system 26
5.2 Multilayer Perceptrons 26
5.3 Recurrent Neural Networks 28
5.3.1 Imbalanced data 30
5.3.2 Cross-validation 30
5.4 Hidden Markov Model 33
5.5 Summary 34

6 Conclusion **35**
6.1 Conclusion 35
6.2 Discussion and future work 36

Bibliography **38**

1 Introduction

The amount of traffic has grown over the past years as more people are driving. This increase in traffic has resulted in an increase in the number of traffic jams and the number of accidents. Accidents are one of the leading causes of traffic jams, with the exception of rush hour traffic, because they block lanes and cannot be solved until the right authorities arrive. The number of deaths and injuries caused by traffic accidents are still a major concern. Fast responses to accidents can save lives and are also able to reduce the amount of time spent in a traffic jam.

Traffic is monitored using different sources of information such as inductive traffic loops and cameras. This information is used for different purposes such as indicating the amount of traffic or to allow people to look at the video feed and determine if an accident has happened. Many existing systems rely on traffic loop data and a rule-based classifier to detect stopped vehicles. However, these systems produce a vast amount of false alarms. Rush hour traffic jams, for example, cause a lot of alarms, but these are not caused by accidents and do not require attention of the traffic controller, so these alarms should not be triggered.

Because of this, traffic controller stopped trusting the existing systems and rely mostly on what they see on the cameras. Humans need time to go through all the camera footage and need some time to respond to the images they are seeing, while a perfect system should be able to respond almost instantly. So by relying on humans instead of an automatic system the total response time increases.

The current rule-based systems are designed to find cars which are having trouble, by giving an alert for every car that is potentially having trouble. The scope of these systems is limited to a per traffic loop basis, so if a car is consistently driving slow then this car keeps creating alarms. Machine learning can be used as an alternative for rule-based systems.

Computer vision can be used to detect abnormal events at intersections[13]. However, the computer vision field is mainly focused on tracking vehicles[3] not on classifying traffic situations. Vast amounts of processing power are needed for real-time processing of video. Processing the data generated by traffic loops using other machine learning techniques requires less processing power. Computer vision

requires using a deep convolution network for every camera of the tunnel with the input with the size of the resolution of the camera. While the size of the data generated by the traffic loops is smaller and convolutions are not required. The data generated by the traffic loops is a sequence of time-series data. Using sequences to detect specific events has shown to be effective using Hidden Markov Models[17] and also using Recurrent Neural Networks[15].

During this thesis an attempt is made to create a machine learning solution to classify multiple different classes of traffic patterns in real-time using traffic loop data. The correct detection of traffic jams and accidents are the most important because they have the most substantial impact on the congestion. A Multilayer Perceptron, Hidden Markov Model and a Recurrent Neural Networks are used for classification. Their results are compared with the performance of an existing rule-based traffic monitoring system.

1.1 Research questions

The main question of this thesis is:

- Is machine learning able to outperform the existing rule-based system?

The sub-questions of this thesis are:

- 1. Which of the following machine learning methods performs the best: Multilayer Perceptron, Recurrent Neural Network and Hidden Markov Model?

Multilayer Perceptrons are a common machine learning method, which could yield good results. However, Multilayer Perceptrons are not specifically designed to handle sequences of data. Hidden Markov Models are able to capture trends in sequence data and should be able to perform accordingly. However, using Recurrent Neural Networks to process sequence data is becoming more popular. This paper compares the results of using each method.

- 2. Can machine learning be used to detect more different classes of traffic patterns than the existing rule-based system?

The rule-based system only tries to find stopped vehicles, but this thesis tries to find more specific classes. A stopped vehicle can mean a number of things: an accident, car trouble, a traffic jam or human behavior. This paper tries to differentiate between

the first three classes. By doing this the traffic controller gets more specific alerts and he or she can make more informed decisions by just relying on the alerts.

- 3. Is a model which is specifically designed for the layout of the tunnel able to outperform a more general model which is not specific for the layout of the tunnel?

Creating a model specifically for one road results in overfitting on the layout and data of that road. By doing this the learned knowledge is specific for the used road and less transferable. The model will only work on other roads with the same amount of traffic loops, lanes and exits. By creating a more general model which is not specific for the layout of the used tunnel the performance could drop, but the model should be transferable to other roads.

1.2 Contributions

The contributions of this paper are:

- A machine learning method which is able to differentiate between normal and abnormal traffic behavior.

The machine learning models trained during this thesis were able to outperform the existing rule-based system when they were used as a binary classifier. They are able to differentiate between normal and abnormal traffic. However, the machine learning models were not able to successfully differentiate between multiple classes.

- A performance comparison between Multilayer Perceptrons, Recurrent Neural Networks and Hidden Markov Models.

The Multilayer Perceptrons were able to outperform the existing rule-based system if a larger sequence of data was used. Which is shown by a F_1score of 0.7872, while the F_1score of the rule-based system is 0.6601. Both the Long Short-Term Memory and the Gated Rectified Unit outperformed the MLP with F_1scores of 0.8181 and 0.8236 respectively. The Hidden Markov Models however, were not able to outperform the rule-based system with a F_1score of 0.0059.

- A performance comparison between a model which is specifically designed for the layout of the tunnel and a more general model which is not specific to the layout of the tunnel.

The Full Road Network classified the traffic patterns at each traffic loop simultaneously for a whole direction of the tunnel. This is done by using all the features of all the traffic loops in that direction together as input for the network. This approach yielded bad results with a F_1score of 0.0231. A more general network is the Traffic Loop Network, which uses the features of the traffic loop that is currently classified and its neighbors. By doing this the network is not depend on the number of traffic loops in the tunnel and is not depend on the layout of the tunnel. The TLN outperformed the FRN with a F_1score of 0.8236.

2 Related work

There is research based on data from inductive traffic loops, most of it is focused on classifying vehicles or estimating the speed of the vehicles. At the moment of writing there is no paper that tries to classify specific traffic behaviors based on the data from traffic loops. However, there are papers that do classification based on time-series data. This section provides an overview of related work of research on traffic loops and research in time-series data classification.

2.1 Inductive traffic loops

A traffic loop exists of multiple wire loops which are installed in the road. The wires are connected to a box, on the side of the road, which powers the closed electrical circuit in the wires. Traffic loops can be used to determine different information about vehicles passing over it. A traffic loop works by powering its wire loops and measuring the decrease in inductance in the wire loops when a vehicle comes in contact with them. The decreases can be measured and based on this measurement a curve can be made over time. Normally a traffic loop exists of two consecutive wire loops with a small distance between them, for example sixty centimeters.

The speed of a vehicle can be determined by using the time a vehicle needs to get from the first wire to the second wire in combination with the distance between those wires. The length of a vehicle can be determined by using the time between the front of the vehicle touches the first wire and the back of the vehicle touches the first wire in combination with the calculated speed.

2.1.1 Speed approximation

Most traffic loops are used in pairs, but Sun et al. presented a method to compute the speed of a vehicle using a single loop[23]. By using linear regression on the waveforms of the single loop they were able to approximate the speed of vehicles in real-time, so it can be used for traffic management. The accuracy of using a single loop is lower than using double loops so for high accuracy double loops are still needed. However, double loops could also be used to calculate the accelerations of vehicles between the two loops. Oh et al. were able to not only detect speed using a

single loop, but also to classify the type of vehicle using machine learning on the data of a single loop[19].

2.1.2 Vehicle classification

Traffic loops are also used to determine the type of vehicle that is driving by. In 2001 Gajda et al. started by detecting two classes a bus and a car using very short loops[6]. Short loops are closer together and because of that, they are able to detect the number of axles and the distance between them. After that Oh et al. were able to detect eight different classes with 82.6% accuracy[19]. Later Jeng et al. used a heuristic decision tree in combination with k-means in order to classify fifteen different classes of vehicles[12]. This method is able to do real-time vehicle classification with a 96% performance rate using a single loop without using explicit axle information.

2.1.3 Accident detection

Traffic loops are able to detect the speed, length and amount of vehicles, because of that they could be able to give an indication whether an accident has happened. Oh et al. published a paper which classified different kinds of vehicles, but more importantly how to classify six different levels of read-end collision risk(from A through F)[18]. The risk is classified using a fuzzy-clustering algorithm, but the algorithm was not tested on data generated by real traffic loops.

Shariat-Mohaymany et al. used the data of traffic loops on a two-lane rural road to identify the significant predictors for an accident[22]. The significant indicators appeared to the Percentage Time Spent Following(PTSF), the percentage of heavy vehicles, the mean speed, the directional distribution of traffic(DDT) direction, the standard deviation of speed and if the section of the road is curved or not. The PTSF is the percentage of time that is spent by vehicles following the same vehicle. The DDT is the distribution of traffic in each direction. Changes in these variables indicate the increased or decreased chance of an accident.

2.2 Sequence classification

There are three dominant categories of sequence classification methods, feature-based classification, distance-based classification and model-based classification[27][1].

2.2.1 Feature-based classification

Most machine learning methods are not able to process raw time-series data directly, in order to use these methods the data has to be aggregated into features.

Throhidis et al. showed that binning data using a histogram could be an effective way to generate discrete features from time-series data[24]. By aggregating the data over a time-period features could be calculated such as the number of beats per minute. Discrete features can be extracted from time series data using the Discrete Fourier Transform(DFT).

Wavelet/Shapelet decomposition has proven to be an effective method to extract features from a sequence[10]. Wavelet transforms express the sequence as the coefficients of a function. The coefficients are calculated at different time resolutions and different locations in time. Güler et al. confirm that calculating the wavelet coefficients could be a good representation of time-series data. They also showed that Lyapunov exponents are a good alternative[9].

After the features are extracted using one of the specified methods they can be used for training and classification using traditional methods. Such as Multi-Layer Perceptrons and Support Vector Machines which generally could not handle time-series data correctly.

2.2.2 Distance-based classification

Distance-based methods define a distance function to measure the similarity between sequences[27]. The performance of distance-based classification is mostly dependent on the performance of the distance function. Euclidean distance is a widely used option, but it requires two time-series to have the same length. Dynamic Time Warping(DTW) aligns two time-series to overcome this problem by minimizing the total distance between the two sequences. The Needleman-Wunsch algorithm can also be used to determine the optimal alignment between two sequences by using dynamic programming.

Once a distance function is defined a number of different methods can be used to determine the class of the chosen sequence. The methods K Nearest Neighbor(KNN) or Support Vector Machine(SVM) can be used when a supervised learning method is possible. If unsupervised learning is needed k-means, Self-Organizing Map(SOM) and Fuzzy C-means(FCM) have successfully been used in the past[1].

2.2.3 Model-based classification

Wang et al. showed that a Recurrent Neural Network(RNN) in combination with and Adaptive Differential Evolution(ADE) algorithm is able to outperform a DTW[26]. The RNN that was used as an echo state network, which has randomly connected internal units in its hidden layer. The ADE is adapted until a specified criterion is reached.

Lipton et al. used a Recurrent Neural Network to diagnose 128 different diagnoses using thirteen sampled clinical measurements[15]. The network is only trained on the raw time series and it outperforms a multilayer perceptron trained on hand-engineered features. The best performing model was a LSTM(Long Short-Term Memory) with dropout and target replication. Target replication is used in order to provide a local error signal at each step.

Hidden Markov Models(HMMs) can be an effective way of multiclass classification[17]. Because of the way HMMs handle temporal data they are also useful for processing time-series data. Lui et al. showed that using multiple HMMs combined could improve the classification performance[16]. It has also been shown that combining a more discriminative model such as an SVM with the HMM improves the performance because they are more potent in classification problems[2][25].

Convolutions can also be used to chain multiple time steps together in order to work directly with sequences[14]. If the input is seen as the bottom layer then the network keeps going from a very wide layer to smaller and smaller layers until the output layer is reached. Overlapping windows of the last layers are used in the next layers in order to find the temporal relationships, this can be seen in figure 2.1. Peddinti et al. achieved a competitive score on different speech recognition datasets using the Time Delay Neural Network(TDNN)[20] which is shown in figure 2.1.

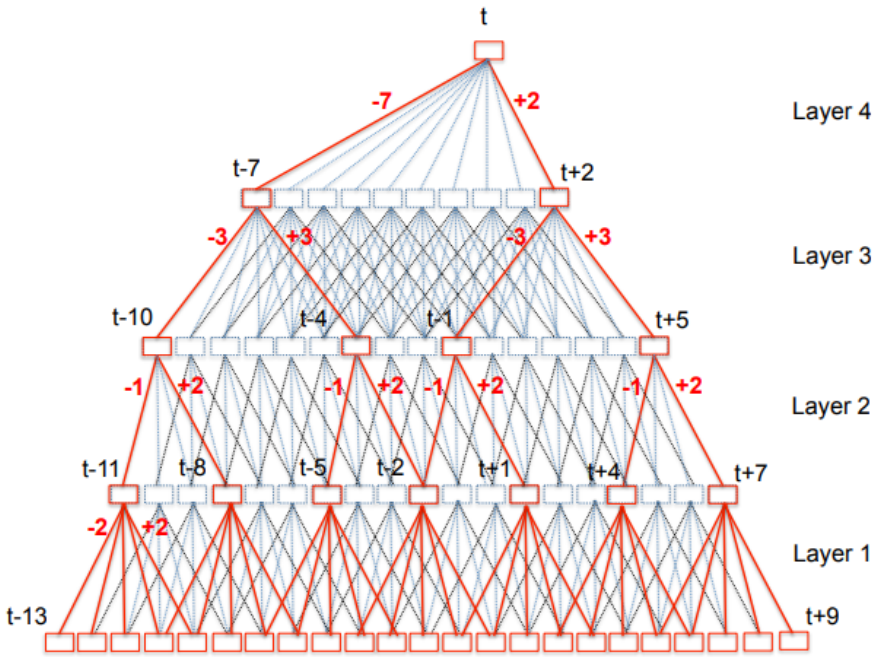


Fig. 2.1: The layout of the Time Delay Neural Network(TDNN) used by Peddinti et al[20].

3 Experimental setup

3.1 Dataset

Data was collected by a partner of Covalent in order to enable this thesis to use machine learning on traffic behavior. Traffic loop data of the Ketheltunnel in the Netherlands was used. The data consists of 53 days of data in which over one hundred million car detections took place. The classifications of the existing rule-based system are also available and will be used as a baseline to determine if the machine learning based approach is able to outperform the existing system.

The rule-based system takes the speed, length and direction of the current car at the current traffic loop into account. The rules can result in four different outcomes: normal traffic, a potentially stopping vehicle, a stopped vehicle and a wrong way driver. Wrong way drivers are always classified correctly based on the direction of the car. So wrong way drivers are not taken into account in this thesis. Cars which are driving slower than fifty kilometers an hour are considered potentially stopping vehicles. Cars which are driving slower than fifteen kilometers an hour are considered stopped vehicles. The outcomes of traffic loops are influenced by their direct neighbors by trying to suppress indicating the same outcome again in the next traffic loop. However, this suppression does not seem to work well in practice.

The system which generates the data is event-driven and an event is generated every time a car passes a traffic loop. The traffic loop data exists of all the car events for every traffic loop. Every car event consists of the speed of the car v, the length of the car l, the direction of the car d, the identifier of the traffic loop the car passed, the time of the car event t and if the information of the car event is reliable. The right side of the tunnel has a total of 205 traffic loops over the whole road and the left side of the tunnel has 183 traffic loops. The maximum allowed speed and other information on the indicators above the road were also available and could potentially be used during experiments to normalize the speed of each car. If cars move a lot slower than the allowed speed then this could be an indication of a specific traffic situation. A list of logged incidents was also available for the time-period of the data. This list exists mostly of people with car trouble. Road traffic controllers have to keep a log of the incidents that they and other authorities notice,

these logs exist of a rough approximation of the time and the closest hectometer sign of the location.

3.2 Visualization

In order to allow for the proper annotation of the data a visualization was made of the Ketheltunnel on which the data was shown. An image of the layout of the tunnel was used as background of the visualization, and this image was split into two halves which are shown underneath each other in order to allow the use of one screen. So the first and third lanes from the top down are both part of the same side of the tunnel which in the case of this tunnel is called the left part of the tunnel and the other two lanes are the right part of the tunnel. The side of the tunnel is also indicated on the right side of figure 3.1. The same parts of the tunnel can also be recognized by the direction of the cars, for the same part of the tunnel they move in the same direction. The direction of the car can be recognized by the orientation of the image of the car.

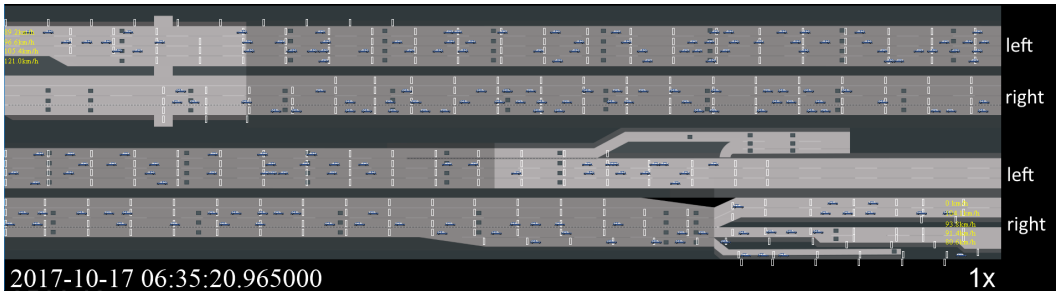


Fig. 3.1: The visualization of the data in the Ketheltunnel. The text on the right side of the image indicates the side of the tunnel.

The traffic loops are drawn as white squares but can be drawn in different colors to indicate different classifications, for example a stopped vehicle is indicated in blue and a traffic jam in orange. Because the flow of traffic is important to understand what happens the visualization is not a still image, but it is comparable to a movie because the cars are moving. The rendering is done at sixty frames per second. The time in the visualization is used to get the correct data and show it in the visualization.

The exact location of a car is only known when it drives over a traffic loop which generates an event in the data. When a car event takes place an image of a car is drawn on the traffic loop associated with that event. The speed v of the car in that event is used to approximate how fast the image of the car should move in the direction of the next traffic loop. Until the next traffic loop is reached no information

is available over the exact position of the car, so the location of the cars between traffic loops is an interpolation.

The speed v is used to calculate after how much time the car should reach the next traffic loop. If this time period has elapsed the image of the car which was associated with that specific event is removed from the screen. Because the speed of cars does not vary a lot between traffic loops a new car event should take place on the next traffic loop shortly after the removal of the car image. This process keeps on happening until a car leaves the tunnel and there are no traffic loops anymore to trigger events.

In order to help the annotator estimate the speed of the cars, the average speed of each lane is shown in kilometers per hour on the sides as yellow text. It is also possible to manipulate the speed of the time in the visualization. The speed is indicated in the right corner of the visualization. The visualization can be paused, slowed down or sped up. Normally the speed of the time in the visualization is real-time, so one second in the visualization takes the same amount of time as one second in reality. However, this can be changed by using specific keys on the keyboard. The final result of the visualization is shown in image 3.1.

3.3 Annotation

These visualizations of the traffic were used by one person to annotate the different traffic situations for all the 53 days. The annotation of the data is done on traffic loop level. The traffic loops are given classes which indicate the traffic behavior between the current traffic loop and the next one. The annotation is done by clicking on a specific traffic loop and selecting the right class according to the person that is annotating. The state of the loops is saved in memory and only the state-changes are saved to the annotations file. So if in an hour only one stopped vehicle is seen there will be two state-changes. Adding the stopped vehicle indication to that specific loop and removing the stopped vehicle indication from that loop. By keeping state, the annotator does not have to readjust the states of each loop for every frame.

The following definitions were determined before the human classification to enforce a consistent framework of reference:

- **Traffic jam**: A queue of cars which for a longer period of time have to drive slower than an average of fifty kilometers an hour. So where $\bar{v} < 50$ and $tp > 5$s.

- **Stopped vehicle**: A vehicle that stops and stays in the same place for at least a minute before it starts moving again.

- **Car trouble**: A stopped vehicle which was has been logged by the road traffic controller.

- **Normal traffic**: Everything which is not defined by the other definitions.

Where $\bar{v}$ is the average speed of all the vehicles between the current traffic loop and the next traffic loop. And where tp is the amount of time specific behavior takes place.

Traffic jams can be recognized by looking for a relatively high amount of cars between two or more traffic loops which are driving slowly for a longer period of time. A traffic jam also has to slow down the cars driving behind it, if the other traffic is not affected by a small queue it is not a traffic jam. Looking at the change of speed of the cars behind a queue gives a good indication if something is a traffic jam. The rear-end of a traffic jam can be recognized by cars significantly slowing down in order to comply with the speed of the cars between the next two traffic loops. For example, if a car has to slow down from 70km/h to 30km/h to be part of a queue then depending on the size and time-window of the queue it is a traffic jam.

Stopped vehicles can be found in the visualization by looking for a car which disappears when driving from loop to loop but does not reappear. If a car reappears within a minute and drives away again then the car is not counted as a stopped vehicle.

The defined traffic behaviors have all been found in the dataset while no accidents took place during the period of 53 days. This means there was no data available to learn to detect accidents. Some road maintenance did take place during the 53 days, the periods where this happened were removed from the dataset.

3.3.1 Data preprocessing

The data which is produced by the event-driven system has to be preprocessed in order for Machine Learning methods to be able to process it. The data was preprocessed by aggregation of the data over a specified timestep ts, in this thesis 10 seconds. The average speed $\bar{v}$, average car length $\bar{l}$ and the number of cars $|c|$ are the features which are calculated for each timestep and for each traffic loop. These features can be calculated by keeping track of all the events for each specific traffic loop in the last timestep. The three features are normalized by dividing them

by the maximum value of that specific feature in the whole dataset. The human classified data is also processed in order to determine the truth values for each timestep of aggregated data. Aggregation results in one file which includes all the features for each timestep and one file which includes all the truth values for the same timesteps.

3.4 Evaluation

3.4.1 Metrics

The rule-based method from the existing system was designed to find stopped cars and potentially stopped cars. Because of that the rule-based system has two generalized classes: normal traffic and stopped cars. In practice the rule-based system also appeared to detect traffic jams. In order to compare the rule-based system with the new methods they are compared using two classes: normal and abnormal traffic behavior. For the rule-based system the abnormal traffic behavior is traffic classified as stopped vehicles. While for the new machine learning methods the abnormal traffic behavior is traffic classified as any non-normal traffic behavior class. The following metrics[8] will be used to evaluate the models on a binary level for each time step for each traffic loop:

Precision: Precision is used to determine the probability that predicted abnormal traffic behavior is actually abnormal traffic behavior.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.1}$$

Recall: Recall is used to determine the probability of successfully predicting abnormal traffic behavior when it takes place.

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

F_1 **score**: F_1 score is used to determine the correctness of the predictions that are made and should have been made and which would result in zero if the algorithm would default to normal traffic behavior.

$$F_1 = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{3.3}$$

TP = True Positives.
TN = True Negatives.
FP = False Positives.
FN = False Negatives.

A true positive is a timestep with abnormal traffic on a specific loop which is correctly classified as abnormal traffic on that specific loop. A false positive is a timestep with normal traffic on a specific traffic loop which is incorrectly classified as abnormal traffic. A false negative is a timestep with abnormal traffic on a specific traffic loop which is incorrectly classified as normal traffic. A true negative is a timestep with normal traffic on a specific traffic loop which is correctly classified as normal traffic.

The machine learning based models should be able to capture different kinds of traffic behavior and so they should be to able classify multiple classes. The important parts of the precision and recall such as true positives are changes in the currently tested class versus the other classes. So a false positive is a timestep classified as class C which true class is a class other than C. In order to measure the performance for multiple classes the following metrics will be used:

Macro F_1 score:

$$MacroF_1 = \frac{F_1 1 + F_1 2 + \cdots + F1N}{N} \tag{3.4}$$

Micro F_1 score:

$$MicroF_1 = \frac{2\text{TP1} + 2\text{TP2} + 2\text{TPN}}{2\text{TP1} + \text{FP1} + \text{FN1} + \cdots + 2\text{TPN} + \text{FPN} + \text{FNN}} \tag{3.5}$$

The micro and macro F_1 scores work in the same manner as the regular F_1 scores, but they are adapted for multiple classes. The Macro F_1 score is the average of all the F_1 scores of each class. If there is an imbalance in the classes this could still be a relatively high score even if the biggest class performs relatively bad. The Micro F_1 score calculates the score based on all the false positives, false negatives and true positives and does not get influenced by class sizes. Which makes it more realistic when imbalanced data is used.

3.4.2 Test set

For the experiments the dataset was split in training set existing of 46 days of data and a designated test set of 7 days of data. The relatively large test set was necessary in order to make sure a whole workweek was included in the designated test set. In

all experiments the precision, recall and F_1score are based on this designated test set.

3.4.3 Cross-validation

In order to validate that the machine learning methods do not overfit on the data of the designated test set cross-validation is on one machine learning method used. In this case eight fold cross-validation is used, so every training set contains 87.5% of the data and every test set contains 12.5% of the data. The data is split based on time, so the first test set is the first 6.6 days and the second test set is the next 6.6 days and so on. Also a new validation set is created for each fold by randomly removing data points from the current fold's test set. By doing this a validation set is created with the same size of approximately fifty batches of data. This is less than 1% of the test set, which is removed in order to create the validation set. This validation set is used to save the best model according to a combination of the recall and the precision during training.

4 Method

The Multilayer Perceptron and Recurrent Neural Network methods were trained for a maximum of two thousand epochs with a learning rate of $1e{-}4$. In order to be able to fit an epoch in memory an epoch exists of a hundred batches with the size of 1024. Also a learning rate decay was used of $1e{-}6$ in order to not divert away from the current local minimum based one or more unrepresentative epochs. The RMSprop optimizer resulted in the best results during the testing phase, so it was used during the training phase. In the same manner as for the cross-validation a validation set was created by randomly removing data points from the test set. These data points were then used to create a validation set with the same size of approximately fifty batches of data. The validation set has the same imbalanced data distribution as the data of the training set, how this is accomplished is handled in section 4.1.1. The validation set is used to save the best performing model during training. The performance was measured by using a combination of the recall and the precision of the model on the validation set.

4.1 Multilayer Perceptron

Two different kinds of Multilayer Perceptrons(MLPs) were used: one big network which stretched over the whole road called Full Road Network and one network on traffic loop level called Traffic Loop Network. The Full Road Network or FRN has as input the three features of all the traffic loops on a particular road for that particular timestep. So for the left side of the tunnel the input size is $3 * 183 = 549$ and for the right side of the road the input size is $3 * 205 = 615$. In this case the Ketheltunnel has two sides, one in each direction and both have there own FRN. The output of the FRN is the probability of each possible class for every traffic loop on the road. So the output size is the number of classes times the number of traffic loops for that specific side. By doing this the network should predict the class of every traffic loop in that side of the tunnel.

The Traffic Loop Network or TLN has as input all the features of the current traffic loop and its neighbors in a specific width and length which results in a grid of features. For example, if the neighbor width and length both are one, then the traffic loops in three by three grid are selected with the traffic loop which is currently

classified in the middle. An example of the grid is shown in figure 4.1 where the traffic loop that is currently classified is shown in blue and its neighbors in green. The input size of the network is the width of the grid times the height of the grid times the number of features which is three. So for a three by three grid the input size is $9 * 3 = 27$. The output of the TLN is the probability of each of the possible classes for the traffic loop that is currently being classified.

The intuition behind using the neighbors of a traffic loop is that traffic moves in a particular direction and the information of speed changes in that direction and changes in the number of cars is needed to make an informed decision. If a vehicle stops on the current traffic loop then cars will not be able to get passed it and the traffic on the loop in front stops or the cars drive over the neighboring traffic loops. If a traffic loop has no neighbors in a particular direction than the features for those nonexistent loops are zeros.

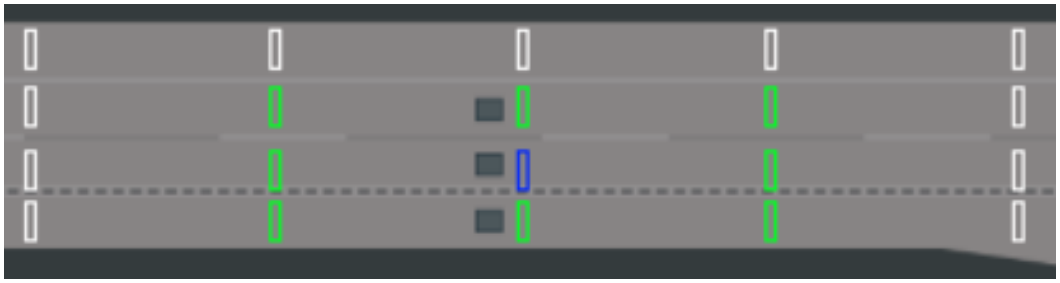


Fig. 4.1: A visualization of the grid with the selected traffic loop in blue with its neighbors in green.

The standard MLP TLN uses the current features of the traffic loops to classify the traffic. However, the data of earlier timesteps is also available so experiments were also done using extra historical timesteps. The MLP TLN which also uses historical timesteps to classify the traffic is called the Historical Traffic Loop Network or HTLN.

4.1.1 Imbalanced data

The dataset is inherently imbalanced because the amount of normal or no traffic is significantly greater than the number of traffic jams and stopped vehicles. For example, in the designated test set the amount of timesteps with normal traffic is approximately twenty-three million timesteps while the amount of timesteps with a traffic jam is approximately thirty thousand timesteps. The split between training and test data was discussed in sections 3.3 and 3.4.3. In order to make sure the networks do not overfit on normal traffic the right division of data has to be found during experimentation.

In order to find the right division, some work has to be done first. The data is divided based on the correct class of the data, so for example the data could be divided into normal and abnormal traffic behavior. By doing this the order of the data points is not chronological anymore. Instead of using the chronological batches every batch is constructed in such a way that the division of the batch is representative for the whole dataset. The division which performed best during experimentation was used during all the training of all the experiments.

MLPs only need the current features of the traffic loops, while Recurrent Neural Networks need a sequence of data. So for RNNs the whole sequence with the size of the chosen number of historical timesteps is saved in the new files. These sequences exist of a sequence chronological timesteps, so in the file of abnormal traffic data only the last timestep of each sequence has to be a timestep with the abnormal traffic. The other historical timesteps of a sequence could be abnormal traffic, but they could also be normal traffic leading up to the abnormal traffic. MLPs could also use the extra data except not explicitly as a sequence. So in this research there is also experimented with MLPs which use a flat version of the same sequence the RNNs use.

The batches used during training have a size of 1024 samples. So if for example two classes abnormal and normal are split 25/75 then 75% normal data would result in 768 samples of normal samples and 256 samples of abnormal data. The amount of samples per class differs significantly. So sampling all samples from a smaller class is realized earlier than sampling all the samples from a big class. When all the samples of a specific class are sampled then the algorithm restarts the sampling of that class from the beginning. By doing this the biggest classes are being undersampled while the smaller classes are being oversampled.

4.1.2 Layouts

The sizes of the layers of both networks were determined during the experimentation phase. The sizes which yielded the best results were used. Adding more than four layers did not result in better results, so no extra layers were used. Increasing the size of the second hidden layer in comparison to the first hidden layer did result in better results.

Full Road Network

The size of the input of the Full Road Network is the number of traffic loops times the number of features which is three. The output is the class of each traffic loop. The layers of the MLPs are fully connected to each other or what is called dense.

Layer	Shape
Input	(Batch Size, |Traffic Loops| * 3)
Dense-ReLu1	(Batch Size, 800)
Dense-ReLu2	(Batch Size, 1200)
Dense-ReLu3	(Batch Size, 800)
Dense-ReLu4	(Batch Size, |Traffic Loops|, 3)
Dense-Softmax output	(Batch Size, |Traffic Loops|, |Classes|)

Tab. 4.1: The layout of the MLP of the best performing Full Road Network.

Traffic Loop Network

The size of the input of the Traffic Loop Network is the width times the height times the number of features which is three times the number of historical timesteps. For the original TLN the number of timesteps is one, while the number of timesteps for the HTLN is six. The output is the predicted class for that specific traffic loop.

Layer	Shape
Input	(Batch Size, width * height * 3 * |timesteps|)
Dense-ReLu1	(Batch Size, 256)
Dense-ReLu2	(Batch Size, 512)
Dense-ReLu3	(Batch Size, 256)
Dense-Softmax output	(Batch Size, |Classes|)

Tab. 4.2: The layout of the MLP of the best performing Traffic Loop Network.

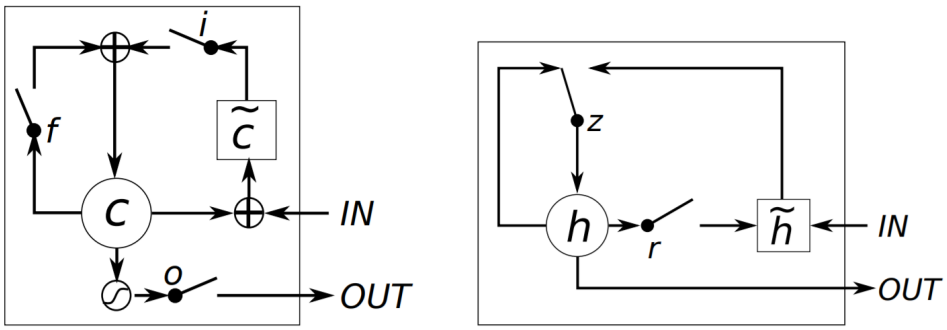
4.2 Recurrent Neural Networks

During experimentation the results of the MLP Full Road Network were below expectations. This is the reason the Full Road Network was not used in experiments with the other machine learning methods.

Multilayer Perceptrons are not specifically designed to capture patterns in data over time, which is important for determining a pattern in traffic. However, Recurrent Neural Networks are designed to use the information from the last timesteps in the sequence in order to predict the class of the current timestep. The input for the RNNs is a sequence of the same data as the MLPs uses as input. In our case the input exists

of six chronological data points of the MLP input. So instead of using one data point of size grid width times grid height times number of features six datapoints are used. These features are then fed to the network in order, the order is important because the RNNs use the activations that happened earlier in later steps. Different types of RNNs reuse earlier activations in different ways, but in all cases they are only used in the hidden layers. In our case the final layer is a fully-connected softmax, which outputs the probability for each possible class. Only the last prediction of the sequence of predictions by the softmax is used, because the intermediate steps in the RNNs do not have old activations yet and are only used to feed the chronological timesteps to the network.

There are different kinds of RNNs and they have shown promising results as discussed in related work. For this research three different types of RNNs were used: Long Short-Term Memory(LSTM), Gated Recurrent Unit(GRU) and Convolution Long Short-Term Memory(ConvLSTM).



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Fig. 4.2: An illustration[4] of both an LSTM and a GRU. In the LSTM i is the input gate, f the forget gate and o the output gate. c is the memory cell and $\tilde{c}$ is the new memory cell. In the GRU r is the reset gate and z is the update gate, while h is the activation function and $\tilde{h}$ is the candidate activation function.

An LSTM[11] is a form of a Recurrent Neural Network and one unit is illustrated in figure 4.2. An LSTM maintains memory in memory cell c in order to be able to learn long-term dependencies. The memory is updated by using the forget gate f to forget part of the memory and by adding the new input from the input gate i. These gates and cells enable the LSTM to decide which information can be forgotten and which new information should be kept in memory. A layer can exist of multiple LSTM units, which each have there own memory cells and gates. The output of these units can then be used as the input of the next hidden layer.

A GRU[4] is also a form of a Recurrent Neural Network and one unit is also shown in figure 4.2. The GRU does not have a specific memory cell, but it uses linear interpolation between the previous activation and the candidate activation to use information from previous timesteps. The update gate u is used to determine how

much the activation has to be updated. A GRU is different from the LSTM in the fact that it cannot control the amount the current state is taken into account when updating the activation. The reset gate r is used to determine how much the GRU should forget the previous state like the forget gate of the LSTM. Multiple GRU units can be used in a hidden layer, just as the LSTM.

The LSTM is able to perform competitively on time series data according to related work. The LSTM does this by learning patterns over short-term and long-term time intervals. The GRU should be able to perform comparably to the LSTM. While also being more efficient because a GRU uses fewer gates and thus fewer calculations are needed. The Convolution LSTM is the logical next step because the data is formatted like a grid and convolutional networks are designed to process grids of data. Convolutional Networks are used for images processing, because they are formatted as a grid of pixels with three dimensions for the color channels.

4.2.1 Comparison with Multilayer Perceptrons

The data of the Recurrent Neural Networks are preprocessed using the same method as the Multilayer Perceptron. The only difference is the fact that multiple timesteps are saved because RNNs use sequences as input. The normal TLN is only trained on one timestep, but the performance of the TLN could also benefit from using extra timesteps. This is why there is also experimented with a HTLN which processes a flat version of the data of multiple timesteps.

4.2.2 Layouts

The layouts of the LSTM and the GRU networks are the same, but with their own respective node. These layouts have been informed by the layouts of the MLPs, but because of the smaller amount of features per timestep less hidden nodes were needed.

Layer	**Shape**
Input	(Batch Size, |timesteps|, width * height * 3)
LSTM-tanh1	(Batch Size, |timesteps|, 64)
LSTM-tanh2	(Batch Size, |timesteps|, 128)
LSTM-tanh3	(Batch Size, 64)
Dense-Softmax output	(Batch Size, |Classes|)

Tab. 4.3: The layout of the LSTM of the best performing Traffic Loop Network.

The convolutional LSTM processes the same data as the GRU and LSTM, but it is formatted in more different dimensions. The convolutional LSTM has an own

dimension for the width, height and the number of channels. For an image the number of channels is the number of color channels and in our case the number of channels is the number of features.

Layer	Shape
Input	(Batch Size, |timesteps|, width, height, 3)
ConvLSTM-tanh1	(Batch Size, |timesteps|, width, height, 64)
ConvLSTM-tanh2	(Batch Size, |timesteps|, width, height, 128)
ConvLSTM-tanh3	(Batch Size, width, height, 64)
Dense-Softmax output	(Batch Size, |Classes|)

Tab. 4.4: The layout of the convolutional LSTM of the best performing Traffic Loop Network.

4.3 Hidden Markov Model

Hidden Markov Models[21] or HMMs are temporal models which use probability distributions over sequences of observations. HMMs observe data in order to find the current state S_t of the current process, in this research traffic in a tunnel. The observations can be discrete or continuous as long as probability distribution can be defined over them[7]. The assumption is made that the observations are made at discrete equally-spaced time intervals, which are 10 seconds during this research.

Another assumption that is made is the Markov property, which states that given the value of the last state S_{t-1} the value of the current state S_t is only dependent on the last state S_{t-1}. The intuition behind the Markov property is that the state at each point in time captures all the historical information that is needed in order to make a prediction. The hidden part of the name Hidden Markov Model comes from the fact that the state path of the process which generates the observations is hidden. This state path exists of the start probabilities of the specified amount of states, the transition probabilities between these states, the covariance and means of the observations in these states and in some cases the emissions probabilities in these states. The score of a state path can be calculated by calculating the log probability of the current observations. This is done by multiplying all the probabilities together and taking the log of it.

A Gaussian Hidden Markov Model is trained by trying to fit the hidden state path and its variables to a given set of observations. Doing this by simply enumerating the possible paths and calculating their scores based on the given observations is only possible for small sets of observations. The Viterbi algorithm[5] is used in order to efficiently find the most probable path. The Viterbi is more efficient by continually calculating the probability of the HMM being in each state after seeing the current sub-sequence of observations. When moving forward through the sequences the values of each historical state are updated recursively with the value of the maximum

probability of all the possible paths to that state. Based on the path with the highest probability the probabilities of the states are then set. The Viterbi algorithm is able to find the most probable path of multiple sequences of observations simultaneously. By doing this the Viterbi algorithm is able to use the same data as the RNNs and find the best parameters for whole batches at a time.

The goal is to find in which state the current traffic loop is at the moment using the same data as the Recurrent Neural Network. As stated in section 3.3 there are four states which can be generalized to the following three states: normal traffic, stopped vehicle and traffic jam. The start probabilities can be calculated by counting the number of times a state happens relative to the total amount of time intervals. The transition probabilities can be calculated by counting the number of transitions between the different states. These values were calculated based on the defined training set and resulted in the model shown in figure 4.3. This model was both used as starting point for the training and as a baseline to compare the results with.

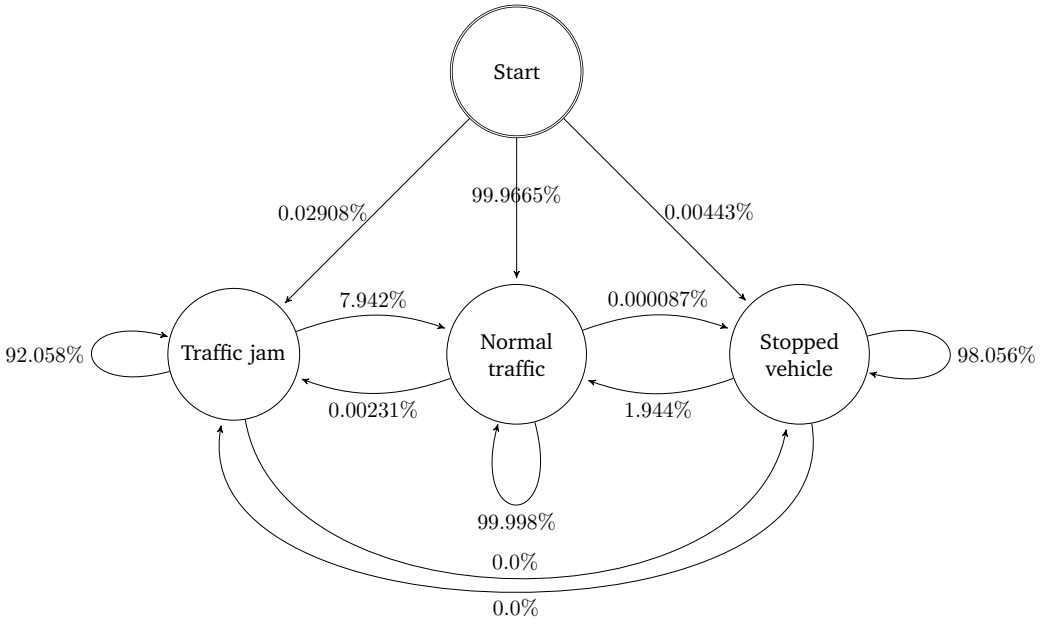


Fig. 4.3: The baseline of the Hidden Markov Model based on the calculated probabilities.

5 Results

5.1 Baseline

5.1.1 A priori probability

In order to prove that the results beat chance the confusion matrix and the evaluation scores are calculated based on the priors of normal and abnormal traffic calculated in section 4.3. In this example one hundred million data points are used. Of a set of one hundred million data points 99.9665% of the data would be normal traffic and the rest would be abnormal traffic. So $100.000.000 * 0.999665 = 99,966,500$ data points are normal traffic and the other $33,500$ data points are abnormal traffic.

If the algorithm classifies 99.9665% of the data randomly as normal traffic and the rest as abnormal traffic then the result is the confusion matrix in table 5.3. The relative amount of true positives is 0.000011% of all the data points. While the relative amount of false negatives and false positives both are 0.033489% of all the data points. These numbers result in very low scores as shown in table 5.4.

	Normal traffic	Abnormal traffic
Normal traffic	99,933,011	33,489
Abnormal traffic	33,489	111

Tab. 5.1: The confusion matrix based on the priors calculated when created the baseline Hidden Markov Model.

	Precision	Recall	F_1 score
Total tunnel	0.00331	0.00331	0.00033

Tab. 5.2: The calculated evaluation metrics using the a priori probability.

If the algorithm classifies all traffic as normal traffic then the amount of true positives would be zero. So this would result in a precision, recall and F_1score of zero. If the algorithm classifies all traffic as abnormal traffic then the recall would be 1.0 because all true positives would be classified correctly. However, the amount of false

positives would be very large. In the example that was used this would result in the following confusion matrix and scores:

	Normal traffic	Abnormal traffic
Normal traffic	0	0
Abnormal traffic	99,966,500	33,500

Tab. 5.3: The confusion matrix based on classifying everything as abnormal traffic.

	Precision	Recall	F_1 score
Total tunnel	0.00034	1.0	0.00067

Tab. 5.4: The calculated evaluation metrics of classifying everything as abnormal traffic.

5.1.2 Rule-based system

Formal specification requires the rule-based system to detect a stopped vehicle within 10 seconds of the event taking place. So the baseline was calculated with $t = 10s$. As stated before the baseline is only able to differentiate between normal traffic and abnormal traffic, so the true positives are correctly classified timesteps with abnormal traffic. The baseline outperforms the a priori chance by a large margin as shown by table 5.5. The rule-based system performs better on the left side of the tunnel during evaluation of the test set, probably because traffic jams on that side of the tunnel have a lower average speed. While the traffic jams on the right side of the right side appear more sudden. The traffic jams on the right side are also finished faster because of the closing of the tunnel which is discussed in section 6.2. Also, the right side of the road is significantly more complicated because of its different exits as can be seen in figure 3.1.

	Precision	Recall	F_1 score
Right side tunnel	0.4951	0.6440	0.5598
Left side tunnel	0.7981	0.6757	0.7318
Total tunnel	0.6561	0.6641	0.6601

Tab. 5.5: The calculated evaluation metrics of the rule-based system on the test set.

5.2 Multilayer Perceptrons

The Full Road Network is not able to capture the difference between normal and abnormal traffic as is shown in table 5.6, but it does outperform the a priori chance. The FRN has more trouble with the left side of the tunnel than with the right side

which is the opposite of the baseline. Using the whole road as input did not result in useful connections between separate parts of the tunnel, but instead resulted in not finding useful connections at all. The network of the right side of the tunnel kept ending in two global minima. In one global minimum the network always predicts normal traffic and in the other global minimum the network predicts abnormal traffic too often.

	Precision	Recall	F_1 score
Right side tunnel	0.0576	0.3255	0.0979
Left side tunnel	0.0124	0.0039	0.0058
Total tunnel	0.0692	0.0139	0.0231

Tab. 5.6: The calculated evaluation metrics of the Full Road Network on the test set.

The Traffic Loop Network is a lot better a capturing the difference between normal and abnormal traffic than the Full Road Network as shown in table 5.7. This could be explained by the fact that some traffic loops in the FRN never have abnormal traffic, so these nodes are never able to predict abnormal traffic. While the TLN learns classification in a generic way for all traffic loops even if they do not have abnormal traffic themselves. The TLN is not able to outperform the rule-based system as shown by the recall and F_1score.

	Precision	Recall	F_1 score
Right side tunnel	0.5449	0.0478	0.0739
Left side tunnel	0.9658	0.5474	0.6987
Total tunnel	0.9315	0.3543	0.5249

Tab. 5.7: The calculated evaluation metrics of the Traffic Loop Network on the test set.

Recurrent Neural Networks and Hidden Markov Models use a sequence to classify data, so effectively they are using more data than the MLP TLN. The MLP TLN was given the same data and it resulted in a performance increase. The TLN is able to find a connection between the historical data and the current data which results in better predictions as shown in table 5.8. The TLN is able to outperform the rule-based system when this extra data is used.

	Precision	Recall	F_1 score
Right side tunnel with extra timesteps	0.5539	0.4987	0.5249
Left side tunnel with extra timesteps	0.9265	0.9339	0.9301
Total tunnel with extra timesteps	0.7989	0.7758	0.7872

Tab. 5.8: The calculated evaluation metrics of the Historical Traffic Loop Network.

Both the TLN and the HTLN network have a significantly better performance on the left side of the road than on the right side, which can be explained by the same reasons as stated in section 5.1. Also after visualization of the predictions, it became clear that the HTLN struggles more with accurate predictions of the first and last rows of traffic loops. This can be explained by the fact that the HTLN has less data there because it cannot get information from the loops in front. Also the two traffic loops with the most traffic jam timesteps on the right side of the road are the loops in the last row, which the network does not handle well. Lastly, the right side of the tunnel gets closed when it gets too crowded, which will be discussed further in section 6.2.

Up until now the networks were only used for binary classification. However, in order to determine if the networks are able to distinguish between multiple classes experiments have been run with three classes: normal traffic, traffic jam and stopped vehicle. Because of the similarity in behavior and the scarcity of the data the classes stopped vehicle and car trouble have been merged into the class stopped vehicle.

	Micro F_1	Macro F_1
Total tunnel with extra timesteps	0.1782	0.3559

Tab. 5.9: The calculated evaluation metrics of the Historical Traffic Loop Network when using multiple classes.

The result of using three specific classes is shown in table 5.9. The extra class decreased the total performance of the HTLN which shown by both the micro F_1 score and the macro F_1 score. The HTLN is not able to find stopped vehicles, which is shown by zero true positives for that class. It also negatively influenced the F_1score which is shown by the macro F_1score in comparison to the F_1score in table 5.8.

5.3 Recurrent Neural Networks

Because of the bad results of the Full Road Network the rest of the results will only focus on versions of the Traffic Loop Network.

The different kinds of Recurrent Neural Networks significantly outperform both the baseline and the Multilayer Perceptron implementations as shown in table 5.10. These results are calculated for the entire tunnel and are based on the classification of either normal or abnormal traffic. The RNNs are able to capture the patterns between the different sequences as the results show it can correctly classify most predictions.

After experimentation the tanh activation function appeared to be the best performing activation function and it was used for all three types of Recurrent Neural Networks. The GRU and the LSTM performed similarly as expected, but the GRU resulted in a better precision while the LSTM had a better Recall. Based on which metric is more important a choice of network could be made, also the GRU is more efficient than the LSTM so that could also be a factor in the choice. Disappointingly the Convolutional LSTM did not result in better or even equal results to the original LSTM. This could be caused by the fact that the grid is too small for the convolutional network to find patterns.

	Precision	Recall	F_1 score
LSTM tanh	0.8334	0.8033	0.8181
LSTM ReLu	0.7135	**0.8812**	0.7885
GRU tanh	**0.8667**	0.7847	**0.8236**
ConvLSTM	0.7332	0.7629	0.7477

Tab. 5.10: The calculated evaluation metrics of the different kinds of Recurrent Traffic Loop Networks on the test set.

As shown in table 5.11 both the LSTM and the GRU have the same problems as the TLN and HTLN. The left side of the tunnel is more straightforward to classify than the right side of the tunnel.

	Precision	Recall	F_1 score
LSTM left side tunnel	0.8966	0.9440	0.9197
LSTM right side tunnel	0.6849	0.5655	0.6195
GRU left side tunnel	0.9059	0.9209	0.9134
GRU right side tunnel	0.7845	0.5517	0.6478

Tab. 5.11: A comparison of the performance of the LSTM and the GRU on both sides of the tunnel.

Using an LSTM to classify multiple classes did not result in finding stopped vehicles successfully. Not a single stopped vehicle was found successfully, which was also the case for the MLP HTLN. Both F_1scores are better than using the MLP to classify multiple classes. However, both the multi-class MLP and the multi-class LSTM were not able to outperform the rule-based system.

	Micro F_1	Macro F_1
Total tunnel with extra timesteps	0.2363	0.4664

Tab. 5.12: The calculated evaluation metrics of the Traffic Loop Network when using an LSTM with multiple classes.

5.3.1 Imbalanced data

As stated in the method the imbalance of the data has to be managed in order to not overfit on a specific class. This is done by changing the relative amount a class occurs in a batch during training. The batches in the designated test set are not changed and are still chronological. After experimenting with different divisions of normal traffic and abnormal traffic the best division for the LSTM Traffic Loop Network appeared to be 99% normal traffic data and 1% abnormal traffic data as shown in table 5.13. This means that for every batch 99% of the batch exists of normal traffic data while the other 1% exists of abnormal training data. This division is representative for the division of normal and abnormal data that exists in the training set. Other distributions resulted in an overfit on the abnormal traffic class as the table shows.

Percent normal traffic	Precision	Recall	F_1 score
50%	0.1747	**0.9321**	0.5534
78%	0.2942	0.9188	0.6065
84%	0.3048	0.8609	0.5829
90%	0.3717	0.8664	0.6191
95%	0.3361	0.9056	0.4903
98%	0.3923	0.9159	0.5493
99%	**0.8334**	0.8033	**0.8251**

Tab. 5.13: The calculated evaluation metrics of the LSTM Traffic Loop Network on the test set with the different class balances.

The division of normal and abnormal data of the LSTM was also used as starting point for the division of the TLN MLP. After experimentation a division of 98% normal traffic and 2% abnormal traffic appeared to give the best results for the MLP.

5.3.2 Cross-validation

Cross-validation was used in order to confirm that the results of the Recurrent Neural Networks were not specific to the current data division between the training set and the designated test set. The results of table 5.14 and the figures show that there are varying differences between the different folds. As shown in figure 5.1 the precision of fold one and three are outliers in comparison with the rest. The amount of abnormal traffic during the test week of fold three was significantly smaller than all the other weeks. The abnormal traffic that did happen in the test set took place at the right side of the tunnel, where all the networks struggle. However, fold one

did have enough abnormal traffic. Most of traffic in fold one took place on the right side of the road, which caused the primary cause of the bad precision the closing of the tunnel. The closing of the tunnel is explained further in section 6.2.

Figure 5.2 shows that fold two is an outlier based on its lower recall. The amount of abnormal traffic in the test set was relatively small and a significant part of the abnormal traffic existed of stopped vehicles. The network has hard time detecting stopped vehicles. Figure 5.3 shows that all the networks have acceptable F_1 scores which are all around the average with no unexplained outliers. The designated test set that was chosen for the experiments is a superset of the best performing fold. This can be explained by the fact that it has a large amount of abnormal traffic, which happens on both sides of the road. The amount of stopped vehicles is small compared to the traffic jams and the right side of the road does not dominate the performance because of the even division between the sides of the tunnel.

Fold	Precision	Recall	F_1 score
1	0.5500	0.7067	0.6186
2	0.8350	0.5517	0.6644
3	0.4814	0.8899	0.6248
4	0.7148	0.8652	0.7828
5	0.7858	0.6233	0.6952
6	0.8079	0.8124	0.8102
7	0.8247	0.7253	0.7718
8	0.8541	0.8345	0.8442
Average	0.7317	0.7511	0.7265
σ	0.1123	0.0994	0.0758

Tab. 5.14: The results of using a GRU to cross validate the data set.

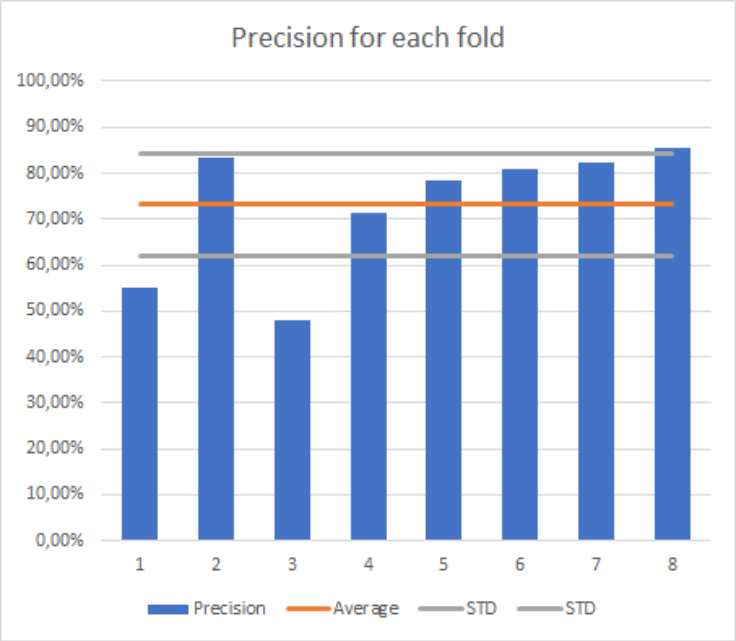


Fig. 5.1: A comparison of the precision of the different folds.

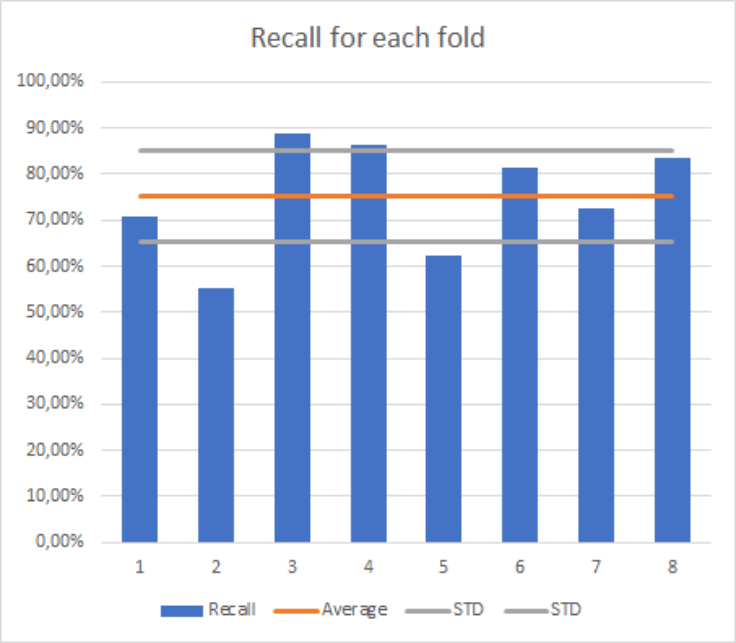


Fig. 5.2: A comparison of the recall of the different folds.

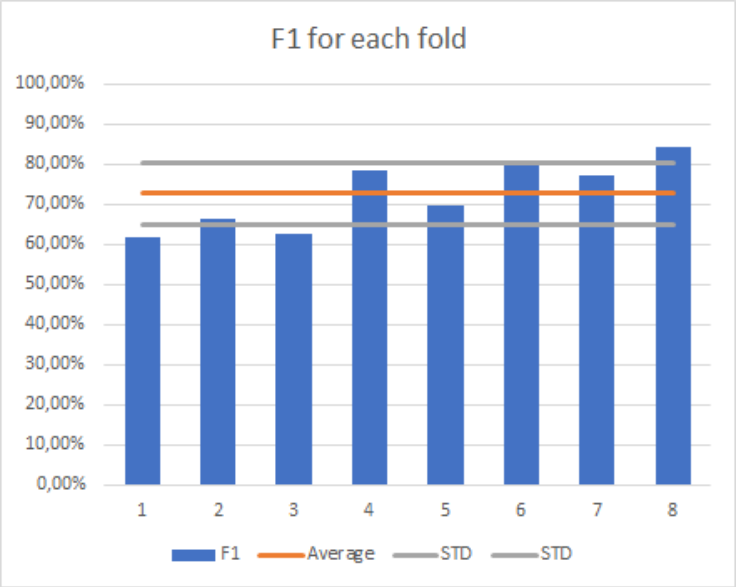


Fig. 5.3: A comparison of the F_1 score of the different folds.

5.4 Hidden Markov Model

The Hidden Markov Model was tested with different types of precalculation. The probabilities from the baseline could be used as precalculated parameters which are not changed during the training. These precalculated parameters do influence the probability during training and thus change learned probabilities during training. A completely unsupervised Hidden Markov Model is not able to capture the three classes or the difference between normal and abnormal traffic. This was expected because it is difficult to find the classes without supervision especially due to the imbalance. The probabilities shown in figure 4.3 are used in order to get the results from table 5.16.

Using the start probabilities informed the Hidden Markov Model about the imbalance in the data and allowed it to train the best Hidden Markov Model. This model beat the a priori probability by a small margin, but it performed worse than the FRN and thus it performed badly. The Hidden Markov Model was not able to find the same imbalance by using precalculated transition probabilities and performed worse. Fixing both the start and transition probabilities by precalculating them results in using the baseline from figure 4.3. The only thing the Hidden Markov Model is able to train at that point is the covariance and the means of the observations for specific states. The results of using both the start and transition probabilities are the worst of all three HMMs. The performance of the Hidden Markov Models was worse than the performance of the Full Road Network. Using two states instead of three did not

result in better a better performance and the Hidden Markov Model was also not able to successfully find a stopped vehicle.

Precalculated	Precision	Recall	F_1 score
Start probabilities	0.9004	**0.0059**	**0.0059**
Transition probabilities	**0.9860**	0.0024	0.0048
Both start and transition	0.9404	0.0021	0.0043

Tab. 5.15: The calculated evaluation metrics of the different kinds of Hidden Markov Models with two states on the test set.

Precalculated	Micro F_1	Macro F_1
Start probabilities	0.0016	0.0025
Transition probabilities	**0.0028**	**0.0048**
Both start and transition	0.0026	0.0017

Tab. 5.16: The calculated evaluation metrics of the different kinds of Hidden Markov Models with three states on the test set.

5.5 Summary

Type	Precision	Recall	F_1 score
A priori chance	0.00331	0.00331	0.00033
Rule-based	0.6561	0.6641	0.6601
FRN MLP	0.0692	0.0139	0.0231
TLN MLP	**0.9315**	0.3543	0.5249
HTLN MLP	0.7989	0.7758	0.7872
TLN LSTM	0.8334	**0.8033**	0.8181
TLN GRU	0.8667	0.7847	**0.8236**
TLN ConvLSTM	0.7332	0.7629	0.7477
TLN HMM	0.9004	0.0059	0.0059

Tab. 5.17: A summary of the calculated evaluation metrics of the different kinds of machine learning methods and the old system.

Conclusion

6

6.1 Conclusion

This thesis proposed a number of different machine learning methods for the classification of traffic patterns. These machine learning methods were used to classify normal and abnormal traffic based on data from traffic loops in a tunnel. The old rule-based system was used as baseline in order to check if the new machine learning based methods are performing better.

The common Multilayer Perceptron or MLP was used to determine if machine learning methods could outperform the old rule-based system. Two different types of MLPs networks were used to classify the current traffic pattern. The Full Road Network or FRN uses the features of all the traffic loops of the whole road in order to instantly predict the traffic patterns of all the hundreds of traffic loops. Another network called the Traffic Loop Network or TLN uses the information of the current loops and its surrounding loops to classify one traffic loop.

The performance of the Full Road Network was significantly below the expectations and was not used in other experiments again. The MLP TLN outpeformed the FRN, but was not able to outperform the existing rule-based network. By using more historical timesteps the Historical Traffic Loop Network outperformed the rule-based system as a binary classifier by almost 20% based on the F_1 score. Which answers sub-question 3 of the research questions in section 1.1 because the HTLN is a more general model while the FRN is specific for the layout of the road. The HTLN only predicts normal and abnormal traffic, which in most cases is a traffic jam. Predicting the following more specific classes was also tested: normal traffic, traffic jam and stopped vehicle. Predicting these three classes did not yield good results and actually worsened the network's capability of finding abnormal traffic. This could be explained by the scarcity of data of classes other than normal traffic and traffic jam. Which answers sub-question 2 of the research questions in section 1.1.

The data generated by the traffic loops can be seen as time-series data. This is why the Recurrent Neural Networks and Hidden Markov Models were used, they are designed to handle temporal data. The results of the Hidden Markov Models were bad and they did not beat the performance of the MLP. Both the Long Short-Term

Memory (LSTM) and the Gated Recurrent Unit (GRU) were able to outperform the MLP as expected. The recall of the LSTM was better, while the precision and the F_1 score of the GRU were better which makes it the best performing network. Which answers sub-question 1 of the research questions in section 1.1.

The imbalance between the different classes in the data started as a problem. If the right division was not used the machine learning methods started overfitting on one of the classes. However, this problem was solved by changing the division of class data in the batches in a representative manner. The size of the normal traffic data is more than a hundred times as big as the size of the abnormal traffic data. This has to be represented during learning otherwise the network starts learning an unrealistic class distribution. For the Multilayer Perceptron the best performing division was 98% normal traffic and 2% abnormal traffic in each training batch. While the best performing division for the Recurrent Neural Networks was 99% normal traffic and 1% abnormal traffic per batch.

6.2 Discussion and future work

The tunnel that was used for this thesis was the Ketheltunnel in the Netherlands. This tunnel was chosen because of a combination of data availability and the complexity of the tunnel. The data of this tunnel was readily available and it has multiple lanes, while the other tunnel of which the data was available only had one lane in each direction. The downside of this tunnel is the complexity for the network because of the exits which can be seen on figure 3.1. Another downside is the fact that the tunnel will be closed if a traffic jam on the right side of the tunnel grows too far into the inside of the tunnel. This is done in order to keep the people safe in case of a fire. This disrupts the natural flow of the traffic and thus influences the data. Especially the Full Road Network is influenced because it tries to learn on the whole situation while the traffic flow keeps getting disrupted.

The currently best performing network should be tested in another tunnel in the future. This could prove that the network is invariant of the tunnel, but this could also prove that a more diverse dataset might be beneficial. More data is also needed in order to determine if the networks could learn the difference between the different classes. The amount of data for the stopped vehicles was too scarce, while using more data could also indicate the fact that the current networks are not able to find stopped vehicles.

The network is currently based on data aggregated over a sequence of car events, which is calculated based on the raw data from the traffic loops. The raw data

generated by the traffic loops was not available during this research, but the car events were. In the future the raw data could be used to train the network directly or different features could be calculated using the raw traffic loop data.

Annotating the traffic situations of the tunnel is a very time consuming and precise task. In this thesis it was done by one person. People make mistakes and timing could be off by the reaction time of the annotator. This is why the data needs to be annotated by more people. A combination of the data of different people could be a much more reliable data set. Mistakes can be found by comparing the data and traffic behavior missed by one person could be found by another. Another possibility is finding a better method to annotate the data.

In this paper a Gaussian Hidden Markov Model was used while some papers use a multinomial Hidden Markov Model. A multinomial Hidden Markov Model could also be researched in the future. But in order to do this the features have to discretized to classes of observations instead of using the currently continuous features. This could be done by counting the number of cars within specific speed ranges.

Bibliography

[1] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. „Time-series clustering–A decade review“. In: *Information Systems* 53 (2015), pp. 16–38 (cit. on pp. 6, 7).

[2] Oya Aran and Lale Akarun. „A multi-class classification strategy for Fisher scores: Application to signer independent sign language recognition“. In: *Pattern Recognition* 43.5 (2010), pp. 1776–1788 (cit. on p. 8).

[3] Norbert Buch, Sergio A Velastin, and James Orwell. „A review of computer vision techniques for the analysis of urban traffic“. In: *IEEE Transactions on Intelligent Transportation Systems* 12.3 (2011), pp. 920–939 (cit. on p. 1).

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. „Empirical evaluation of gated recurrent neural networks on sequence modeling“. In: *arXiv preprint arXiv:1412.3555* (2014) (cit. on p. 21).

[5] G David Forney. „The viterbi algorithm“. In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278 (cit. on p. 23).

[6] Janusz Gajda, Ryszard Sroka, Marek Stencel, Andrzej Wajda, and Tadeusz Zeglen. „A vehicle classification based on inductive loop detectors“. In: *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*. Vol. 1. IEEE. 2001, pp. 460–464 (cit. on p. 6).

[7] Zoubin Ghahramani. „An introduction to hidden Markov models and Bayesian networks“. In: *International journal of pattern recognition and artificial intelligence* 15.01 (2001), pp. 9–42 (cit. on p. 23).

[8] Cyril Goutte and Eric Gaussier. „A probabilistic interpretation of precision, recall and F-score, with implication for evaluation“. In: *European Conference on Information Retrieval*. Springer. 2005, pp. 345–359 (cit. on p. 14).

[9] Inan Guler and Elif Derya Ubeyli. „Multiclass support vector machines for EEG-signals classification“. In: *IEEE Transactions on Information Technology in Biomedicine* 11.2 (2007), pp. 117–126 (cit. on p. 7).

[10] Qing He, Zhi Dong, Fuzhen Zhuang, Tianfeng Shang, and Zhongzhi Shi. „Fast time series classification based on infrequent shapelets“. In: *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*. Vol. 1. IEEE. 2012, pp. 215–219 (cit. on p. 7).

[11] Sepp Hochreiter and Jürgen Schmidhuber. „Long short-term memory“. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 21).

[12]Shin-Ting Jeng and Stephen Ritchie. „Real-time vehicle classification using inductive loop signature data“. In: *Transportation Research Record: Journal of the Transportation Research Board* 2086 (2008), pp. 8–22 (cit. on p. 6).

[13]Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. „Traffic monitoring and accident detection at intersections“. In: *IEEE transactions on Intelligent transportation systems* 1.2 (2000), pp. 108–118 (cit. on p. 1).

[14]Martin Längkvist, Lars Karlsson, and Amy Loutfi. „A review of unsupervised feature learning and deep learning for time-series modeling“. In: *Pattern Recognition Letters* 42 (2014), pp. 11–24 (cit. on p. 8).

[15]Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. „Learning to diagnose with LSTM recurrent neural networks“. In: *arXiv preprint arXiv:1511.03677* (2015) (cit. on pp. 2, 8).

[16]Kui Liu, Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. „Multi-HMM classification for hand gesture recognition using two differing modality sensors“. In: *Circuits and Systems Conference (DCAS), 2014 IEEE Dallas*. IEEE. 2014, pp. 1–4 (cit. on p. 8).

[17]Sabyasachi Mukhopadhyay, Sanket Nandan, and Indrajit Kurmi. „Multiclass Classification of Cervical Cancer Tissues by Hidden Markov Model“. In: *arXiv preprint arXiv:1512.06014* (2015) (cit. on pp. 2, 8).

[18]Cheol Oh, Seri Park, and Stephen G Ritchie. „A method for identifying rear-end collision risks using inductive loop detectors“. In: *Accident Analysis & Prevention* 38.2 (2006), pp. 295–301 (cit. on p. 6).

[19]Seri Oh, Stephen Ritchie, and Cheol Oh. „Real-time traffic measurement from single loop inductive signatures“. In: *Transportation Research Record: Journal of the Transportation Research Board* 1804 (2002), pp. 98–106 (cit. on p. 6).

[20]Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. „A time delay neural network architecture for efficient modeling of long temporal contexts“. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015 (cit. on pp. 8, 9).

[21]Lawrence R Rabiner. „A tutorial on hidden Markov models and selected applications in speech recognition“. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286 (cit. on p. 23).

[22]Afshin Shariat-Mohaymany, Ali Tavakoli-Kashani, Hadi Nosrati, and Andisheh Ranjbari. „Identifying significant predictors of head-on conflicts on two-lane rural roads using inductive loop detectors data“. In: *Traffic injury prevention* 12.6 (2011), pp. 636–641 (cit. on p. 6).

[23]Carlos Sun and Stephen G Ritchie. „Individual vehicle speed estimation using single loop inductive waveforms“. In: *Journal of Transportation Engineering* 125.6 (1999), pp. 531–538 (cit. on p. 5).

[24]Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. „Multi-Label Classification of Music into Emotions.“ In: *ISMIR*. Vol. 8. 2008, pp. 325–330 (cit. on p. 7).

[25]Michel F Valstar and Maja Pantic. „Combined support vector machines and hidden markov models for modeling facial action temporal dynamics“. In: *International Workshop on Human-Computer Interaction*. Springer. 2007, pp. 118–127 (cit. on p. 8).

[26] Lin Wang, Zhigang Wang, and Shan Liu. „An effective multivariate time series classification approach using echo state network and adaptive differential evolution algorithm“. In: *Expert Systems with Applications* 43 (2016), pp. 237–249 (cit. on p. 8).

[27] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. „A brief survey on sequence classification“. In: *ACM Sigkdd Explorations Newsletter* 12.1 (2010), pp. 40–48 (cit. on pp. 6, 7).