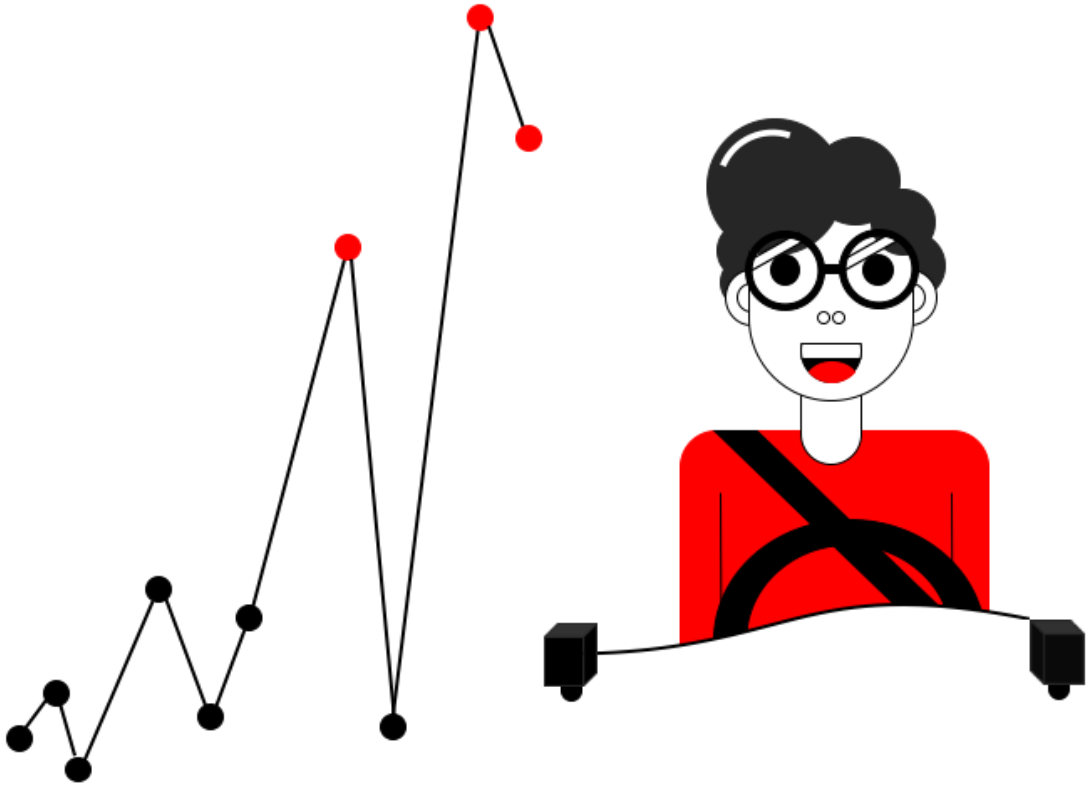

Anomaly detection of driver behavior

Deep machine learning and statistical methods are investigated to detect anomalies in driver behavior

Master's thesis in System, control and mechatronics

KLAS SVENSSON QVISTBERG
PONTUS BJÖRKMAN

Department of Electrical Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Anomaly detection of driver behavior

Deep machine learning and statistical methods are investigated to detect anomalies in driver behavior

KLAS SVENSSON QVISTBERG
PONTUS BJÖRKMAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Anomaly detection of driver behavior
Deep machine learning and statistical methods are investigated to detect anomalies in driver behavior

KLAS SVENSSON QVISTBERG
PONTUS BJÖRKMAN

Supervisor: Torsten Wilhelm, Smart Eye AB
Examiner: Tomas McKelvey, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Image showing a person driving a car and a graph illustrating the collected data from the driver.

Typeset in LaTeX
Gothenburg, Sweden 2021

Anomaly detection of driver behavior
Deep machine learning and statistical methods are investigated to detect anomalies in driver behavior

KLAS SVENSSON QVISTBERG
PONTUS BJÖRKMAN

Department of Electrical Engineering
Chalmers University of Technology

Abstract

In recent years, deep machine learning and statistical methods used to detect anomalies has grown. In this project, statistical methods based on nearest neighbor and Mahalanobis distance algorithms are used with different model configurations to detect anomalies in a driver's behavior. Also a method based on deep machine learning approach is implemented in the form of a transformer network.

The definition of an anomalous driver behavior is a hard line to draw, and in this project it has been regarded as a behavior or condition causing reduced awareness of the road. Data consisting of mostly normal driving behavior was used to train the different models, where Smart Eye's tracker core system had been previously used to extract important features from the data. To validate the results, different anomalous scenarios were recorded, including e.g. sleeping, vomiting and fighting, which were then fed to the implemented models to evaluate how well these could be identified as anomalies.

The models were all implemented with and without the addition of an accumulating error, acting as an added form of temporal information for the models. This was done to investigate the importance of temporal information in the given context, as well as trying to make the models more robust overall. The parameters of each model were optimized by first looking at their results when applied to a subset of normal behavior.

All models performed well when it came to not reporting false positives during normal scenarios, while the transformer network performed the best at identifying anomalies. The added temporal aspect did not significantly improve the performance of the statistical models, however it is theorized that with better optimized parameters for the temporal algorithm this should yield better results.

Keywords: Deep machine learning, statistical methods, transformer, kNN, Mahalanobis distance, anomaly detection, anomalies, outliers, driver behavior

Acknowledgements

We want to show gratitude to everyone that helped us during the Master Thesis project. Specially thanks to our supervisor at Smart Eye AB, Torsten Wilhelm, who have contributed with a lot of inputs during the whole project. Secondly we want to thank Henrik Lind for the opportunity to collaborate with Smart Eye AB and all the knowledge he has shared to us. We also want to thank Victor Brandt and Simon Duvevik for their help during the project. Lastly we want to thank Tomas McKelvey, our examiner, for giving us the opportunity to do this Master Thesis project.

Klas Svensson Qvistberg & Pontus Björkman
Gothenburg, May 2021

Contents

List of Figures

List of Tables

1
Introduction

The following chapter provides an introduction and motivation to understand the main goals and motivations for this thesis project. It also introduces Smart Eye AB, which is the company this project was created in collaboration with.

1.1 Background

Data science is a field of study which aims to extract patterns from sets of available data, in order to draw conclusions about general behaviors within the data set. While data science dates back all the way to the 17th century [1], the huge advancements in computing power and storage capacity of modern computers has revolutionized the field. In 2005 Roger Magoulas coined the term 'Big Data', to describe data sets with enormous amounts of data [2]. Huge progress has since been made within the field of data science to find methods which can be applied to extract knowledge from these highly complex data sets. These methods range from purely statistical approaches, to machine learning methods, where the subfield deep learning has been utilized to great success in the recent decade. Deep learning methods are designed to mimic the workings of the biological neural network of the human brain. This is done by designing artificial neural networks, which consist of multiple layers of artificial neurons, which are connected in different patterns depending on the task it is applied to [3].

One field of data science consists of identifying abnormal events within data sets, namely anomaly detection. This is the process of detecting deviating items or events in a data set, which otherwise display a general pattern [4]. Depending on the properties of the data set being investigated, such as univariate vs. multivariate data or anomaly detection in time series data, different methods have been found to excel. This will be elaborated on in Chapter 2 of this thesis.

1.1.1 Smart Eye

The project being proposed was formed together with Smart Eye AB. The company was founded in 1999 and develops artificial intelligence based eye tracking technology, especially within the automotive industry. The eye tracking technology predicts, assists and understands human actions and intentions. Besides the automotive industry, Smart Eye AB is also involved in industries such as aerospace, aviation and clinical research [5].

Smart Eye can also offer different products in different areas which includes Driver Monitoring System (DMS), Applied Artificial Intelligence systems (AIS) and research instruments. All these products are developed to reduce the total amount of accidents in the traffic. This is the key to why Smart Eye was founded and has grown under the last decade and continue to grow [5].

Customers are located all over the world and has developed eye-tracking solutions to clients such as NASA, Harvard University, Tonji University, BMW and Geely [5].

1.1.2 Anomaly detection

Anomaly detection is a hot area in data analysis and means that rare items, observations or events are identified, which can be seen as outliers in a dataset. In other words, data points which are significantly deviating from the rest of the data points. An example of two outliers can be seen in Figure 1.1. In this example, it is assumed that knowledge about the data states that the values should lie in the interval $y = [-10, 10]$ for the normal process.

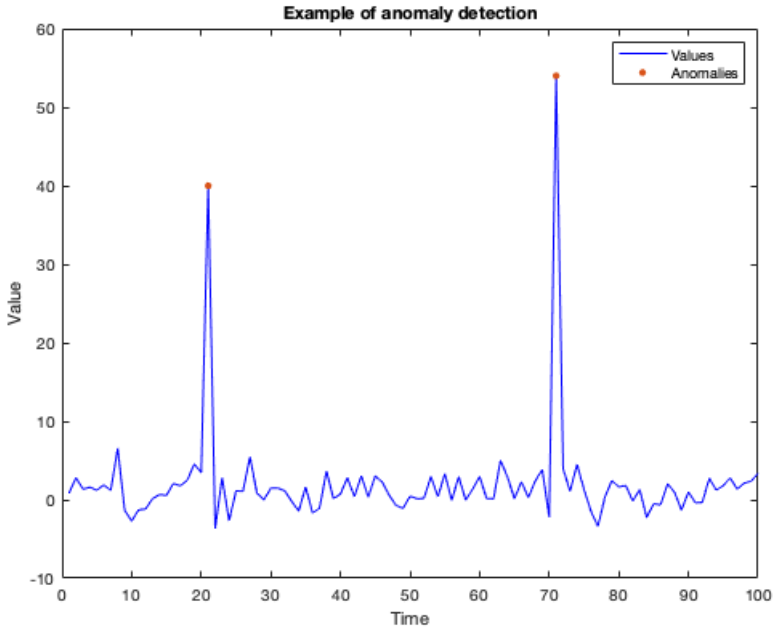

Figure 1.1: Example of anomaly detection in a dataset. The anomalies can be vizualized at timestep $t = 20$ and $t = 70$.

Anomaly detection can be applied to a number of fields, and a lot of companies utilize it within e.g. economics and cyber security. The example illustrated in Figure 1.1 is an anomaly detection problem of a single feature (value in this case) changing over time. However, most problems are more complex and often include multiple features, which themselves can be correlated to each other with varying degree.

There exist a multitude of methods for detecting anomalies, where statistical analysis and machine learning methods are the two most common approaches in modern anomaly detection. The aim for a statistical analysis approach, when considering an anomaly detection problem, is to analyze given data using different kinds of data manipulation methods. This often includes investigating properties of the data such as its distribution, mean and variance, which can be used in different methods such as Z-score or histogram analysis.

The other branch of common methods is machine learning. What type of machine learning method is being used depends on the type of problem, but include e.g. support vector machines, autoregressive models or neural network models for classification. One way to utilize machine learning models for anomaly detection is to have labeled data of anomalous events, which can then be learned explicitly by the model. However, due to the very definition of an anomaly, these events are often sparse and hard to detect, which means that having a plethora of labeled data with anomalies may not be a feasible approach. A more common method is instead to train models using unsupervised learning in for example an autoregressive fashion. Here, a model is fed data which is assumed to include (almost) only "normal" data, i.e. with a minimal amount of anomalies. The model then learns the underlying pattern of the process being investigated and can then recreate the signal being investigated. When the model later is deployed and fed data of an anomalous event, this will lead to a larger reconstruction error due to the fact that the model hasn't been trained to reconstruct that particular behavior, and the anomaly will be identified.

1.1.3 Related work

There are papers available about anomaly detection with different techniques. Two different methods to detect anomalies in datasets are deep machine learning and statistical methods, which will be investigated in this project. When it comes to methods based on deep machine learning, the majority of the neural networks are trained with unsupervised learning. Papers based on unsupervised learning with a high accuracy on anomaly detection can be seen in [4], [6], [7] and [8]. But there are also other methods used by others with good accuracy. In paper [9], Q. Jan, J. Chen and L. De Strycker proposed a method to detect outliers based on the Mahalanobis distance for source localization, the result shows that the Mahalanobis distance method had the lowest root mean square error (RMSE) compared to the other methods. Other papers that had used the Mahalanobis distance method had obtained surprisingly good results in terms of a small amount of false positive and at the same time detect the anomalies. Other methods used for anomaly detection is Nearest Neighbor (NN) methods. In the paper [10], X. Gu, L. Akoglu and A. Rinaldo showed that NN-methods can result in a good performance to obtain outliers, this was made on an unsupervised learning anomaly detection set-up.

1.2 Project description

The aim for this thesis is to investigate different methods for classifying driver behavior, and in doing so also detecting anomalies in said behavior. The methods being investigated will range from deep learning to statistical methods, and whether or not existing applications for driver monitoring can be utilized will be considered. The following research questions were chosen to illustrate the main pillars of the study.

Are deep learning or statistical methods suitable for detecting unusual driver behavior considering performance and execution time?

There is no information available for this sort of problem, but there are similar studies available. In [11], machine learning has been used to detect anomalies in human poses with good results. This was done by studying the skeleton trajectories. In this project, statistical methods and machine learning models will be trained on a dataset provided by Smart Eye, and then be tested on a new dataset containing labeled events, in order to compare the different approaches.

Can driver anomaly detection be based on spatial information alone or does temporal information need to be exploited for robustness?

To answer this question, multiple models will be developed where temporal information is both included and omitted. This will allow the results to show if temporal information seems to increases the performance or not when included in the algorithms.

Are output data provided by driver monitoring and cabin monitoring systems sufficient for detecting anomalies in driver behavior?

By using statistical methods and machine learning on data provided by driver monitoring and cabin monitoring, this will be investigated by developing algorithms for the detection of anomalous behavior.

Can driver anomaly detection be accomplished by unsupervised learning or are labeled events necessary?

The machine learning- and statistical methods will be trained using unsupervised learning. To evaluate the results, a validation set with labeled events will be created, in order to establish the ground truth during the evaluation.

What can be defined as anomalies in driver behavior?

Firstly, it is hard to define what an anomalous behavior is since there are no general rules explaining this concept. However, to be able to evaluate the investigated methods, a definition for such anomalous scenarios will need to be found. This will be further investigated during the project, where for example the drivers attention on the road as well as physical condition will be regarded as parameters affecting this definition.

1.3 Limitations

To detect anomalies of driver behavior, a lot of data had to be collected on the drivers. To do this effectively and in a safe environment, the data was collected in a garage. During these recordings, the car is stationary and the drivers are provided with instructions to follow during the recordings, e.g. look straight ahead or look to the side. One of the limitations of this project was therefore that the scenarios were staged, which can not be compared to drivers that actually drive a car in a real environment.

Another limitation for the project was that anomalies of driver behavior had to be defined. These anomalies can be defined differently from person to person. During this project, anomalous driver behavior was defined by doing research about e.g. whats happening to the body and head when a disease occurs.

The datasets provided by Smart Eye included a lot of features, everything from points on the head to points on the body. One limitation of the project was that only a few of these features that were deemed most relevant to use for detecting anomalies were included in the different models.

1.4 Thesis Outline

The next chapter explains the theory about the methods implemented. It includes the statistical methods Mahalanobis distance and kNN, and a method based on deep machine learning in the form of a transformer network. The architecture of the transformer is visualized with an explanation of each part. In Chapter 3, the methodology during the project is described, including e.g. how the data is pre-processed, how the methods are implemented to detect anomalies and how the evaluation was made. The results of the project can be seen under Chapter 4, where all the results during the project are stated in graphs and tables. At the end of the report, conclusion and discussion can be found including suggestions how to continue working with anomaly detection of driver behavior.

2
Theory

In this section the theory for the implemented methods are presented. It includes the statistical methods and machine learning in general. Methods that are used for this study are presented in more detail, e.g. the Mahalanobis distance, kNN and a transformer network. The different methods are about collecting, evaluating, presenting and analyzing data or information.

2.1 Statistical methods

Under this section the theory about the Mahalanobis distance and the kNN are presented. The performance of these different statistical methods depend on the dataset in terms of distribution, variance, mean value and if there are disturbances from the collection etc.

2.1.1 Mahalanobis distance

The Mahalanobis distance is similar to the Euclidean distance which is a common method to obtain outliers in a dataset. The Euclidean distance calculates the shortest possible distance between two points if and only if the scalings are the same in the dataset. If the scalings are different (common if different features are included), equal weights are assigned to these variables. This will result in additional weight for the single characteristic and the variables that are correlated gets a higher weight than it should [13].

The Mahalanobis distance method solves this problem. This method was developed by the Indian statistician named PC Mahalanobis [12]. This method takes the correlations between variables into account. The Mahalanobis distance is a scale invariant metric and calculates the distance between a point $x \in \mathbb{R}^p$ generated from a given probability distribution $f_x(.)$ with the associated mean value $\mu = E(\mathbf{X})$ of the distribution [13]. The associated covariance matrix is given by

$$\Sigma = E[(\mathbf{X} - E(\mathbf{X}))(\mathbf{X} - E(\mathbf{X}))^T]. \tag{2.1}$$

Assuming that the covariance matrix has full rank, the Mahalanobis distance formula can be defined as

$$D(\mathbf{X}, \Sigma, \mu) = \sqrt{(\mathbf{X} - \mu)^T \Sigma^{-1} (\mathbf{X} - \mu)}. \tag{2.2}$$

The Euclidean distance can be calculated from Equation (2.2) by applying the identity matrix I instead of the covariance matrix Σ. Further, the sample mean is

calculated as

$$\bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_i \tag{2.3}$$

where x_i is all the specific elements in a row or a column of a dataset. The sample cross covariance can be calculated as

$$C(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})^T}{n-1} \tag{2.4}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are random vectors, $\bar{x}$ and $\bar{y}$ are the sample mean values and n is the number of samples. It describes the relation between two random variables and how they vary together.

Assuming that the parameters Σ and μ describing the distribution of the stochastic variable X are known, the Mahalanobis distance values will result in a chi-square distribution with k degrees of freedom (total number of samples). The shape of the chi-square distributions with respect to k can be seen in Figure 2.1. Note however that if the sampled mean and covariance is used, this will not be the case, but should converge towards this provided an increasing amount of samples.

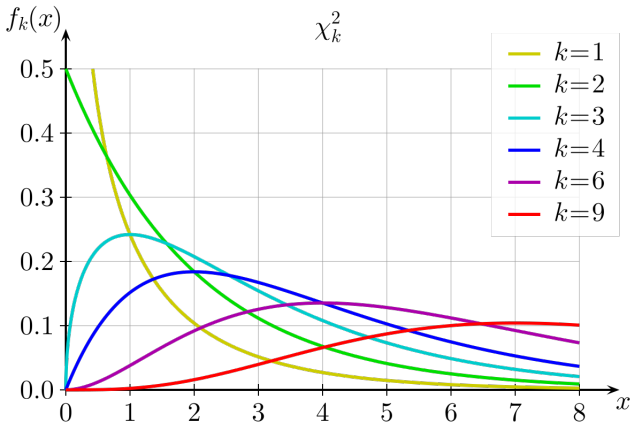

Figure 2.1: Chi-square distributions with respect to degrees of freedom k, taken from [14].

Depending on what the distribution looks like, outliers can be obtained (lies far from the rest of the points). This can be done effectively by defining a critical value. If a Mahalanobis distance point lies above this value, it can be seen as an outlier. There are predefined tables for the critical values depending on k and the probability level. It can be seen in Table 2.1.

Table 2.1: Critical values for a chi-square distribution, taken from [15].

k	0.05	0.01	0.001
1	3.841	6.635	10.828
2	5.991	9.210	13.816
3	7.815	11.345	16.266
4	9.488	13.277	18.467
5	11.070	15.086	20.515
6	12.592	16.812	22.458
7	14.067	18.475	24.322
8	15.507	20.090	26.125
9	16.919	21.666	27.877
10	18.307	23.209	29.588

where k is the number of degrees of freedom (number of samples). 0.05, 0.01 and 0.001 are the probability levels that separates the normal data from outliers, in other words the probability level p_{level} can be set according to Equation (2.5).

$$P(D^2 > \epsilon) = p_{level} \tag{2.5}$$

where ϵ is the threshold value and p_{level} is the probability level.

2.1.2 k-Nearest Neighbor method

Methods belonging to the k-Nearest Neighbor (kNN) class of algorithms is another relatively common method for anomaly detection. A kNN algorithm attempts to extract patterns from a given dataset, e.g. by assuming that data points belonging to a given class (for a classification problem) are situated in close proximity to each other. Therefore, for a given dataset, a new data point can be classified by finding the k nearest neighbors of this new data point in the previously known dataset [16].

The previously described approach does however assume that our dataset consists of a number of known classes, which in an anomaly detection application is not necessarily a feasible assumption to make. This is because anomalies are by definition rare and often unknown occurrences in the data, which are hard to map out due to these reasons. In [17], Mahsa Mozaffari and Yasin Yilmaz propose a kNN-based method called Online Discrepancy Test (ODIT). This method is implemented using a dataset X_N, where N is the number of data points in the dataset, which is assumed to contain nominal, multivariate data only, in other words non-anomalous data points for the system being considered. Data points can then be classified as either normal points or outliers, by comparing these new points to the predefined dataset. This comparison requires that a number of parameters are determined for the method, and these include the number of neighbours k being regarded for the method, the significance level α and the split ratio of the dataset. The entire methodology of the algorithm is presented in the following section.

The dataset is then divided into two subsets X_{N_1} and X_{N_2}, where $N_1 + N_2 = N$

(In [17] this was chosen as a 30/70 split). The reason for this step is solely to increase the computational efficiency of the method. The training phase is then conducted by considering every data point $\mathbf{x}_m \in X_{N_1}$ and computing the euclidean distance $d_i = ||\mathbf{y}_i - \mathbf{x}_m||$ between this data point and all data points $\mathbf{y}_i \in X_{N_2}$, where $i = 1, 2, ..., N_2$. After this a total distance metric L_m is computed for the data point $\mathbf{x}_m$ using the k nearest neighbors of $\mathbf{x}_m$ in the dataset X_{N_2} (corresponding to the k smallest values of d_i computed for the data point $\mathbf{x}_m$) according to

$$L_m = \sum_{i=k-s+1}^{k} d_i^{\gamma} \tag{2.6}$$

where $s \in [1, k]$ is a number defining how many of the k largest nearest neighbors should be considered, and $\gamma > 0$ is the weight. After this has been done for every data point $\mathbf{x}_m \in X_{N_1}$, the vector L_m contains the total distance metric defined in Equation (2.6), for all points $\mathbf{x}_m$ where $m = 1, 2, ..., N_1$. In the next step, the points $\mathbf{x}_m$ corresponding to the K smallest distances in L_m are chosen to model the distribution of points which will be considered corresponding to non-anomalous data in the later implemented test phase. K is chosen using a significance level $\alpha \in (0, 1]$ according to $K = \lfloor N_1(1 - \alpha) \rfloor$.

The test phase is then implemented to handle new data points $\mathbf{x}_t$. The first step of the test phase is to compute the total distance metric L_t of the data point $\mathbf{x}_t$ with respect to the previously used subset X_{N_2} according to Equation (2.6). The value of L_t is then compared to the modeled distribution from the training phase, by considering the largest value from the distribution, namely $L_K = L_m(K)$, according to

$$D_t = k(log(L_t) - log(L_K)) \tag{2.7}$$

where k is the number of features in the process being considered. The value L_K essentially acts as a border between anomalous and non-anomalous values of the data points in this context. If $log(L_t) > log(L_K)$ then Equation (2.7) will take a positive value, meaning the contribution from that data point will be considered a potential anomaly. However, to make the model more robust against sudden spikes in the data due to e.g. noise, the model accumulates an anomaly score Δ_t after every input of new data points $\mathbf{x}_t$. To decide when this anomaly score reaches a level where the input gets flagged as anomalous, a threshold h is defined, and the entire algorithm for these two steps are displayed in Algorithm 1.

Algorithm 1: Algorithm for the computation of temporally accumulating anomaly score

Result: Accumulated anomaly score
initialization; $\Delta_0 = 0$
while *not at end of sequence* **do**
  $\Delta_t = \Delta_{t-1} + D_t$
  $$\mathrm{T(t)} = \begin{cases} 0, & \text{if } \Delta_t < h \\ 1, & \text{otherwise} \end{cases}$$
  if $|\Delta_t| > 2h$ **then**
    $\Delta_t = 2h * \mathrm{sgn}(\Delta_t)$
  else
    pass
  end
end

In Algorithm 1, T signals whether or not time step t is considered an anomaly, which takes the value zero if Δ_t is below the threshold h and the value one if it is above the threshold (signaling that time step as an anomaly). The check $|\Delta_t| > 2h$ makes sure the accumulated error does not diverge too much during long periods of times of only anomalies or normal data. A benefit of this approach is that the temporal aspect of the data is taken into consideration, as the anomaly score is accumulated over multiple time steps. This means that the threshold parameter h can be tuned in such a way that data points from multiple consecutive time steps need to take a positive value D_t before the algorithm considers the sequence anomalous.

2.2 Deep machine learning

Machine learning is a branch of the field of artificial intelligence, which aims to utilize collections of data in order to get a computer to identify patterns from and solve problems associated with said data. A sub field within machine learning is deep learning. Deep learning models consist of algorithms which are constructed to replicate the simpler models concerning the behavior of the human brain, and are commonly referred to as artificial neural networks (ANN) [18]. In its simplest form, an ANN consists of an input layer, one or several hidden layers, and an output layer. Each of these layers are built up of an arbitrary number of artificial neurons, which is where the connection to the human brain is apparent, as these are inspired by the biological neurons in the brain. The ANN is then formed by sequentially connecting the layers, such that each neuron in a layer receives input from every neuron in the previous layer, and then produces an output which is sent to every neuron in the following layer, as illustrated in Figure 2.2.

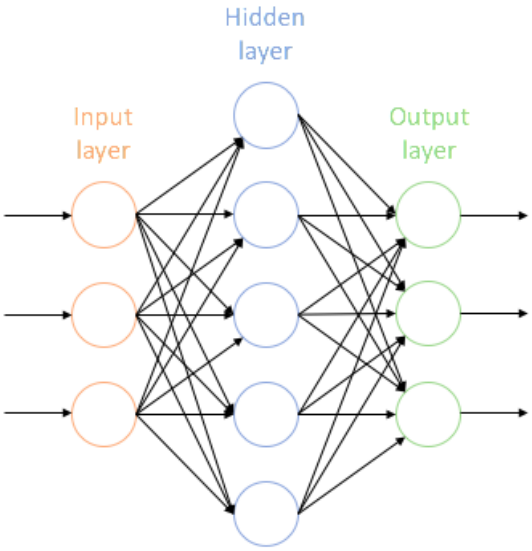

Figure 2.2: Feed forward neural network.

For every input neuron $n_{i,j}$ from a previous layer (where i is the index of the layer and j is the index of the neuron within that layer), each neuron $n_{i+1,k}$ in the next layer (where k is the index of the neuron in the next layer $i+1$) essentially performs the computation

$$f(x) = \sigma(wx + b). \tag{2.8}$$

In Equation (2.8) σ is some activation function, x is the input from the previous neuron $n_{i,j}$, w is the weight of the connection between the two neurons, and b is the bias of the neuron $n_{i+1,k}$. To allow an ANN to also produce non-linear mappings, each neuron applies an activation function to construct the output. There exist many different types of such activation functions, which can be used depending on the desired range of values of the output, but a few common ones include ReLU, sigmoid and softmax [19].

Finally, to train a neural network a cost function appropriate to the task needs to be defined. The value of this cost function is the metric which is used to procedurally optimize the ANN through training. An example of a cost function is the mean squared error (MSE), which is calculated according to

$$MSE_{loss} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{2.9}$$

where x are the observed values of the variable being predicted, $\hat{x}$ are the predicted values and n is the number of variables being predicted. This particular cost function is well suited for regression problems. To train the network, a backpropagation algorithm is then deployed, which computes the partial derivatives of the loss function with respect to all the different weights w in the network. These weights are then updated in order to minimize the loss function, with the aim to yield a better prediction from the network in the next training iteration.

2.2.1 Transformer

The transformer is a fairly novel neural network architecture which has grown in popularity in recent years. It was first introduced in 2017 by Vaswani et al. in the paper "Attention is all you need" [20]. In [20], the transformer network was introduced by applying it to a natural language processing (NLP) problem in the form of language translation. Since then, the potential areas of application for the transformer network has rapidly increased. In 2019, OpenAI used transformers in their language models, and it was also used by DeepMind in their famous AlphaStar model, which is a very complex model trained to play the popular game Starcraft [21].

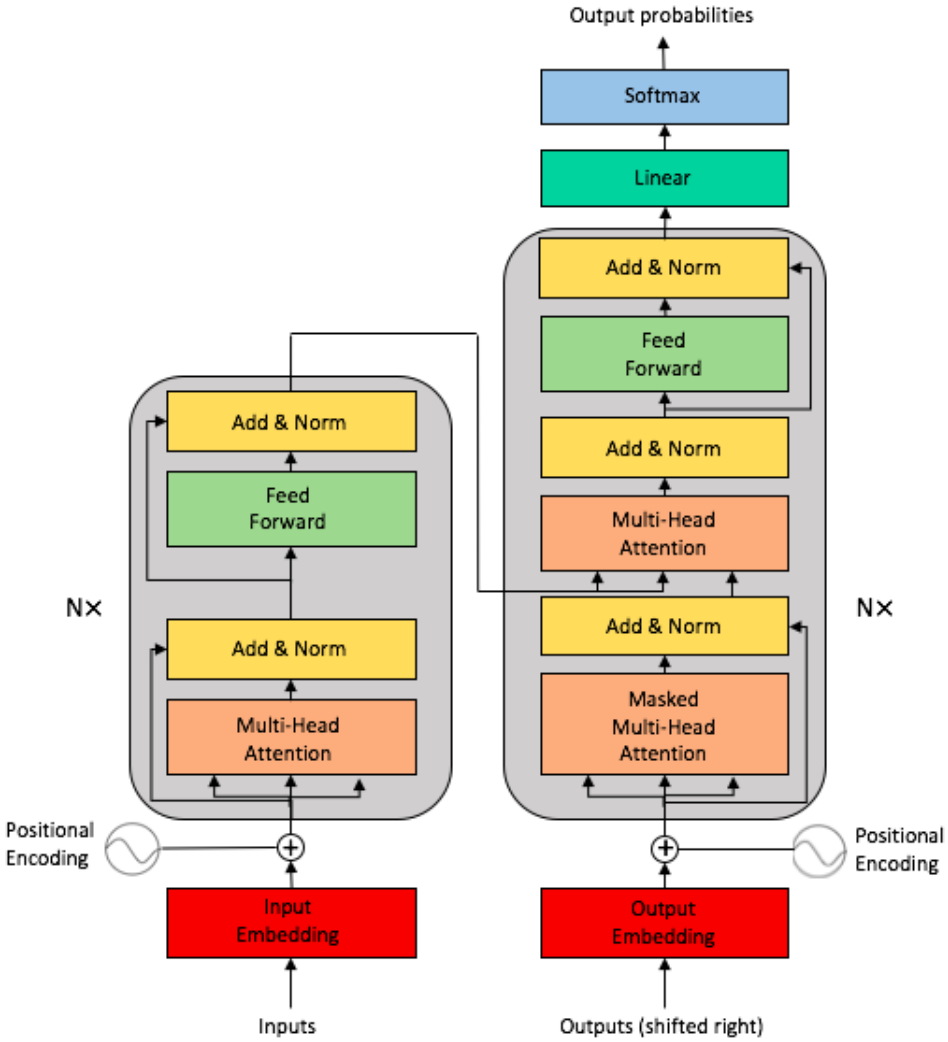

Figure 2.3: Transformer model architecture, inspired from [22].

In Figure 2.3 the architecture of the transformer network is visualized. The transformer is mainly built up by two types of blocks, the encoder block and the decoder block. The encoder block, in the left part of the figure, consists of a multi-head attention layer followed by a add and norm layer with residual connection from the input to the encoder. The output from the add and norm layer is then sent to a

simple feed forward network, which is then sent to another add and norm layer along with residual connection from the previous add and norm layer. A transformer generally consists of multiple encoders in a row, before the output then is sent to the decoder part. The residual connections are therefore of importance in order to prevent the potential problem of vanishing gradients, due to the long backpropagation path during training.

The multi-head attention layer is the unique part of the transformer network. As can be seen in Figure 2.3, the multi-head attention layer takes three inputs, which are called the keys, queries and values of the input to the encoder layer. These three entities are constructed by applying linear mappings W to the input x of the encoder, which generates the keys K, queries Q and values V according to

$$K = W_K x, \tag{2.10}$$

$$Q = W_Q x, \tag{2.11}$$

$$V = W_v x. \tag{2.12}$$

The input x is regarded as a series (often correlated in time) with T steps, and each step is constructed according to a d-dimensional input embedding (which will be explained further on), which means $x \in \mathbb{R}^{d \times T}$. The linear mappings are by extension of size $W \in \mathbb{R}^{d \times d}$, and K, Q and V are resulting matrices of size $\mathbb{R}^{d \times T}$, where each column is the key, query and value for each step in the sequence. These mapping matrices W contain the first set of learnable parameters in the attention layer. The next step in the attention layer is to calculate the attention between the different steps in the input sequence. This is done by treating the previously mapped entities as queries for the key-value pairs. The attention is computed by taking the dot-product between all queries and keys, scaling the result by a factor $\frac{1}{\sqrt{d}}$ (to allow the attention to be independent of the size of the embedding dimension d), and lastly applying a softmax function to gain the weights used in the attention calculation. The attention between different steps in the sequence is then finally computed by taking the dot product between these weights and the values V, according to

$$\textit{Attention} = \textit{softmax}\Big(\frac{QK^T}{\sqrt{d}}\Big)V \tag{2.13}$$

In order to further improve the attention mechanism, Vaswani et al. propose the use of so called multi-head attention. This essentially means that the keys, queries and values are split up into equally big parts, and the attention computation in Equation (2.13) is computed for each part individually. Each of these computations is regarded as a "head" in the attention layer, which is why it's called multi-head attention. Vaswani et al. argues that this leads to a better performance due to the fact that each head can focus on learning different patterns in the sequential input data. For this to work however, the number of heads must be chosen so that the embedding dimension d is evenly divisible by the number of heads.

Finally, the input to the encoder depends on the problem it is being applied to, but has for the most part included sequential data of some sort. As was mentioned

earlier, it has already been proven to excel in NLP problems, and a typical structure of the input in such problems is via one-hot-encoded vectors, describing the index of a word in a predefined vocabulary. This input is then transformed using an embedding layer, where each entry in the input sequence is mapped to the desirable embedding dimension d, used as input to the encoder. The process of mapping the raw input to the embedding dimension varies depending on the problem, and can either be done via a pre-defined lookup table or as a learnable layer in the transformer.

The architecture, as described up until now, is independent of the sequential order of the elements in the input sequence, due to how the attention is computed. To fix this problem, so called positional encoding is added to the input embedding before being sent to the encoder. There are multiple ways to do this, but Vaswani et al. used a method where they use sine and cosine functions of different frequencies to add information of the relative position of a time step. Each element in the input embedding has a term added to it according to

$$PE_{pos,2i} = sin(pos/10000^{2i/d}) \tag{2.14}$$

$$PE_{pos,2i+1} = cos(pos/10000^{2i/d}) \tag{2.15}$$

where pos is the relative position of the element in the input sequence, and i is the index of the embedding within every step in the sequence.

As can be seen in Figure 2.3 to the right, the transformer also includes a decoder layer. The particular transformer illustrated here is the one used for language translation in the original paper. As can be seen the decoder is similar in structure to the encoder, with an additional layer added at the bottom of the decoder, called masked multi-head attention. This is in essence just a multi-head attention layer, but the attention in this layer is restricted to only allow each step in the sequence to regard steps which are situated earlier in the sequence. For machine translation, this is to prevent the decoder to gain information from later parts of the sentence during training, as the entire translated sequence is fed to the decoder during this process. The output from this layer is then fed to an add and norm layer as previously described, and after this comes an additional multi-head attention layer. In this step however, the attention layer regards both the output from the encoder stack along with the output from the previous attention layer in the decoder. The rest of the decoder is similar to the encoder, and (in this particular application) the final output of the transformer consists of the probabilities concerning which word in the vocabulary is most likely to come next.

2.2.2 Transformer for feature value prediction

The transformer model for this project was implemented in a similar way as in [23]. When the transformer is implemented in this way, the decoder part is not necessary. The full transformer architecture has mainly shown state of the art results when applied to machine translation tasks. In that case the encoder computes a continuous attention based representation as in Equation (2.13) (called $\mathbf{z}$ from now on) of an input sentence in one language, then the decoder interprets this representation $\mathbf{z}$ and

recreates it in another language. When the model instead is implemented to predict future values based on an earlier sequence, only the attention based representation **z** is needed, as described in [23]. so for this project, the architecture consists of a 1D-convolutional layer, an attention module and a number of fully connected layers. The architecture is illustrated in Figure 2.4.

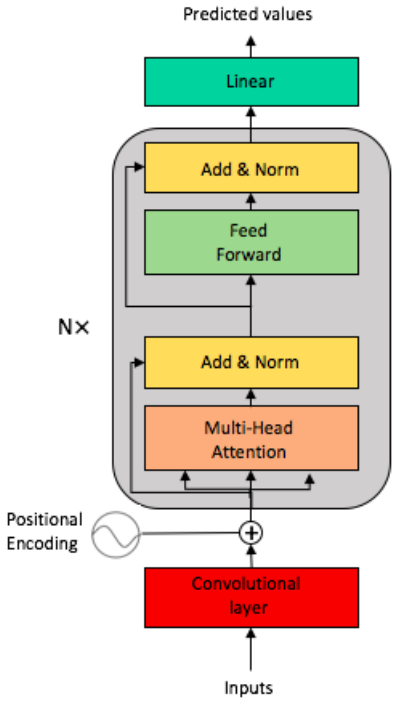

Figure 2.4: Transformer model architecture for anomaly detection, inspired from [20].

The goal of this architecture is to take an input consisting of a time series of length T with a number of features n measured from the system being regarded. In other words the input to the model is a matrix of size $T \times n$. The embedding layer is then formed as a 1D-convolutional layer, which has two main purposes. It creates the input to the encoder of size $T \times dim$ (where $dim > n$), to allow the encoder layer to find more complex patterns in the input sequence. The 1D-convolutions are all applied over all features in a single time step t, which has the added benefit of this embedding procedure being able to find patterns between the features in a single time step, while the encoder then also finds patterns between different time steps in the sequence. Before this is fed to the attention module, positional encoding is added to the input embedding. As in the original transformer paper [20], sinusoidal positional encoding was used as in Equation (2.14) and (2.15). Finally, the output from the attention module is sent through a number of fully connected layers, which reduces its dimension down to the number of features being regarded, and produces the prediction for the values of the features in the next time step.

3
Method

This chapter presents the methodology of the project. The dataset provided by Smart Eye is described, and also the process of gathering evaluation data for the developed methods. The implementations of the different models is also described in detail.

3.1 Data

In order to develop different methods, Smart Eye provided data specifically collected for this type of project. The data had been gathered using a setup of multiple cameras in a car where drivers drive from Gothenburg to Varberg. The multiple cameras manages to capture the driver from different angles in order to maintain the tracking of features such as head pose and gaze direction as accurately as possible.

The dataset provided for this project contained a multitude of features, and was provided in csv-files. An example of how the dataset was structured can be seen in Table 3.1.

Table 3.1: Extracted from one of the datasets.

trackingState/headPose/position/0	**trackingState/headPose/position/1**
0.023464558646082878	0.024783603847026825
0.02351059578359127	0.02479688636958599
0.02355637587606907	0.02480410784482956
0.023591306060552597	0.024805156514048576
0.02362889051437378	0.024803197011351585
0.02365558035671711	0.024796176701784134
0.023692168295383453	0.024794792756438255

All the features used during the project can be seen in Table 3.2. These were used both separately and together, in order to create several different model configurations for every method used.

Table 3.2: Features used during the project.

Features
Head rotation x
Head rotation y
Head rotation z
Head position x
Head position y
Head position z
Consensus gaze direction x
Consensus gaze direction y
Consensus gaze direction z
Upper mouth opening x (Pixel Position)
Upper mouth opening y (Pixel Position)
Lower mouth opening x (Pixel Position)
Lower mouth opening y (Pixel Position)

The head position is measured in a Cartesian world coordinate system, where the origin is situated based on different landmarks in the car's interior. The gaze direction is given as a vector originating in the eye of the driver and describing the direction the driver is looking. Head rotation is described using a rotation matrix, where the features used describes the rotations around the three main axis of the world coordinate system. Finally, the tracking of the mouth features are made using mouth points pinned to different regions of the mouth of the person being tracked. These mouth points are then measured using pixel position in the different cameras being used during the tracking.

3.1.1 Collection of validation data

To evaluate the statistical models and the transformer model, a definition of an anomalous behavior needed to be established. Two main parameters were defined to begin with. The first one concerned the drivers apparent focus on the road, and whether or not he/she seems distracted. The second one, which also affects the focus of the driver, concerned the drivers physical and mental condition. Based on these two parameters, a number of anomalous scenarios were defined and recorded, which can all be seen in Table 3.3.

Table 3.3: All anomalous scenarios recorded for validation.

Scenario number	All scenarios recorded
1	Look behind the shoulder
2	Drunk
3	Taking off the jacket
4	Heart-attack
5	Fight
6	Panic-attack
7	Shocked/Scared
8	Change playlist on phone
9	Vomit
10	Sleep
11	Dead

A document was created with descriptions of each scenario and can be seen in Appendix 1 and was used as a reference if the person did not have ideas how to perform the act. For the scenarios concerning acting of different health related conditions, the description was defined based on typical symptoms which are present during such an event. 15 different persons did the recordings to get a more accurate evaluation instead of evaluating on just a few ID:s.

The recordings were done in a garage with a stationary car. Two different camera systems were used, both a 2-camera system and a 5-camera system based on Smart Eye AB:s own product Smart Eye Pro. The training data used was saved from the 5 camera systems where two of the cameras mounted on the dashboard were used to save the data. To make the passages where the anomalous scenarios were performed more visible, the test subject was asked to pretend to drive "normally" in between the scenarios. A tool could then be used to annotate the first and last frame of every scenario during each recording. This was done by instructing the subjects on when each scenario could be started in real time, and then determining when the scenario was finished. A timeline for every recording can be seen in Figure 3.1.

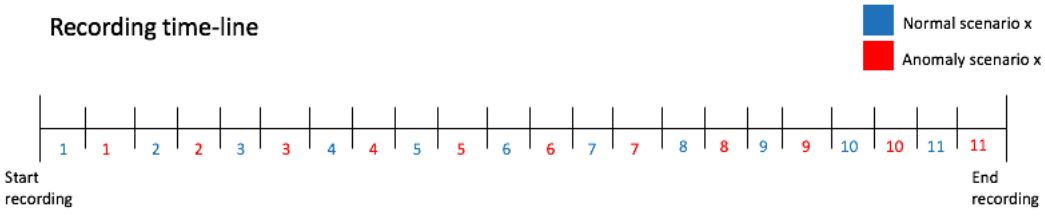

Figure 3.1: Timeline how the normal- and anomalous scenarios are structured under all recordings.

3.2 Trade-off in anomaly detection methods

For all the model configurations there is a trade-off between the ability to detect anomalies and the risk of reporting false positives. This is determined by the threshold value ϵ, which defines the boundary between if a frame is considered an anomaly or not, based on some metric depending on the method used. With an increasing threshold value ϵ, the amount of anomalies reported will decrease, however so will the number of reported false positives (incorrect classification of normal data). On the other hand if ϵ is decreased, this will result in more anomaly classifications, including both true positives (correct classification of anomaly) and false positives. This threshold value was defined for the different methods based on the assumption that the majority of the training data contains normal driving behavior. This meant that when the validation of the methods was performed, sequences in the validation data which deviated significantly in the metric being used (e.g. Mahalanobis distance, kNN-distance etc.), and could also be well explained (when looking at the corresponding video frames) based on the definition of an anomaly from the previous section, was used to define these threshold values ϵ.

3.3 Mahalanobis

This section presents the methodology regarding the implementation of the Mahalanobis distance method, how the data was pre-processed and which features were used. In the end the methodology of the evaluation is presented, e.g. how to detect the anomalies, both in the dataset and the corresponding frame.

3.3.1 Pre-processing of data

Since there were a lot of features in the data obtained from Smart Eye, relevant features were picked. The features initially picked included head position, gaze direction and head rotation. The quality of each feature is measured in every time step t and can also be picked from the dataset. The quality values are in the range $[0,\ 1]$, where 1 is perfect tracking and 0 means the tracking is lost. When the tracking is lost, the values of the corresponding features load as NaN. When this was the case, the data point including one of these NaN values had to be removed. This was made by iterating over every feature vector and check every value. If there was a NaN value at the current time step t, that data point had to be deleted for the current time step. The next step was to define a threshold β, to delete all the values that correspond to bad tracking $x_q < \beta$, where x_q is the quality for a data point x. The features were also applied in different combinations, both together and individually.

Since the data was divided into different *.csv* files for each person ID and run, a *.yaml*-file was created with all the paths noted. The *.yaml*-file was then loaded to the main-script.

3.3.2 Implementation

After the pre-processing of data was made, the implementation of the Mahalanobis distance was started. It can be divided into two parts, one implementation to calculate the mean and covariance of all the features chosen with a big amount of data, and one implementation for the validation, in other words load the saved mean and covariance and calculate the Mahalanobis distance for a new dataset that should be evaluated.

The mean value was calculated for every feature vector by using Equation (2.3) on the majority of the data. The sample mean values were then saved in a vector as

$$\bar{\mathbf{x}} = [\bar{x}_1 \ \bar{x}_2 \ \cdots \ \bar{x}_n]^\top \tag{3.1}$$

where n is the number of features used. The sample covariance was then calculated on the same dataset by using equation 2.4. The sample covariance results in an estimate of the covariance matrix Σ according to Equation 2.1. For the Mahalanobis method, the data used to calculate the mean values and covariance matrix included recordings of 30 different test IDs p_{total}, where 10 csv-files f containing approximately 5000 data points d each were used. This results in approximately

$$P = p_{total} f d = 1\ 500\ 000 \ \ points. \tag{3.2}$$

This amount of points were used because at this point the sample mean and covariance had converged up to the tenth decimal point, which was deemed sufficient. Also, the use of multiple test IDs makes the mean and covariance more robust when it comes to different drivers having slightly different "normal" driving behavior. As was mentioned in the pre-processing, covariance and mean were also calculated both with and without quality pruning. The quality threshold $\beta = 0.1$ was used for this, which resulted in the number of data points used for the pruned case was decreased by approximately 10%.

The mean and covariance was then saved as a csv file, which could later be loaded to another file to calculate the Mahalanobis distance formula on new validation sets, including both data from IDs that was included in the training process and ones that were not included. In the validation sets, between $3 - 6$ data files were used at the same time, which corresponds to $15\ 000 - 30\ 000$ data points. The loaded mean and covariance and the collected validation data was then inserted into the Mahalanobis Equation (2.2). The points were then analyzed to see if any outliers could be obtained in a dataset of mostly normal driver behavior and is explained under 3.3.3 Performance test.

3.3.3 Threshold tuning

The threshold tuning for the Mahalanobis distance can also be divided into different parts, all based on analyzing the data to obtain outliers. This was done by plotting the Mahalanobis distance of every frame over time, and then defining a threshold for the critical value ϵ, where all frames with a Mahalanobis distance greater than

this threshold is considered an outlier.

The first part of the evaluation was to plot the Mahalanobis distance in a histogram to see the distribution of the data. In theory, the Mahalanobis distance formula converts a normal distribution of data to a chi-square distribution, and depending on the application area and the total amount of features used (see Figure 2.1), the distribution looks differently. An example of the distribution of the data with head position, gaze direction and head rotation as features in a histogram can be seen in Figure 3.2.

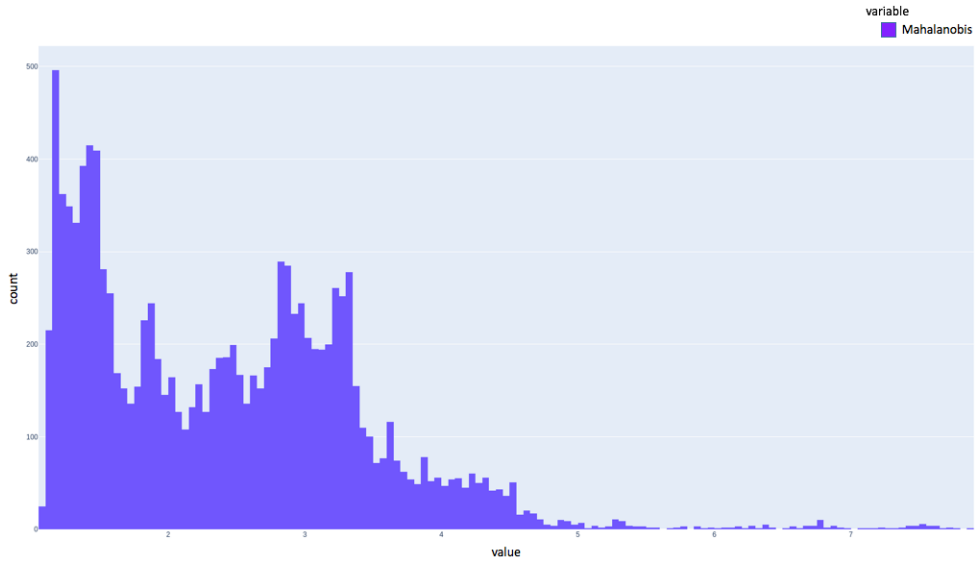

Figure 3.2: Distribution of the data after the Mahalanobis distance calculations in Histogram. Validation-set on mostly normal data.

Depending on which data was included in the validation set, the histogram plots followed the chi-square distribution with varying accuracy. The reason for this variance was because of the varying amount of bad quality data $x_q < \beta$ in different validation sets. When a feature is tracked with a low quality, its measurement defaults to zero, which in the computed Mahalanobis distance metric results in a measurement $D \approx 3$, which explains the "bump" in the distribution seen in Figure 3.2, causing it to diverge from the chi-square distribution. By investigating a number of distribution plots, the threshold value $\epsilon = 5$ was chosen with trained features on head position, head rotation and gaze direction. To further investigate the different validation sets, the calculated Mahalanobis distance was plotted for every frame over time. In Figure 3.3 and 3.4, the data from the earlier distribution plot is illustrated for both mean and variance computed on all available data, as well as quality filtered data.

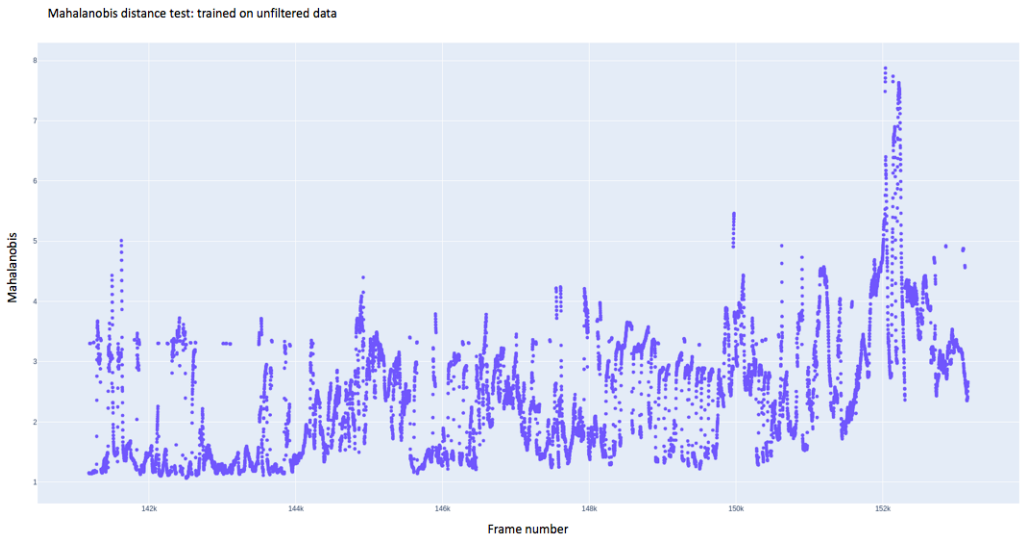

Figure 3.3: Scatter-plot of the data after the Mahalanobis distance calculations. Validation set on mostly normal data.

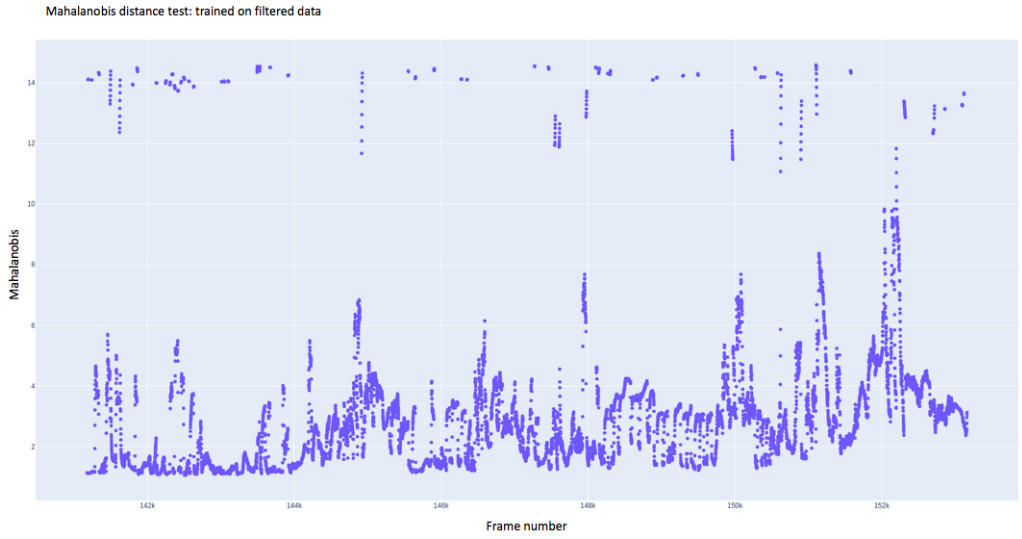

Figure 3.4: Scatter-plot of the data after the Mahalanobis distance calculations with filtered data, $\beta = 0.1$. Validation set on mostly normal data.

By comparing the scatter-plots for the filtered and unfiltered data, a "belt" of points was obtained in Figure 3.4 around $D \approx 14$. This occurs since the validation part includes all data and the training part to calculate the covariance and mean are based on the filtered data in this case. The points in this belt corresponds to data points that have bad quality. Because of this, going forward it was decided to use mean and covariance obtained without filtering the data.

Further more, by investigating the plot in Figure 3.3, a clear peak above the threshold $\epsilon = 5$ can be seen at the end of the sequence. To investigate what actually happens at this point in the video, a script was made for loading the corresponding

frames from this part of the video. An example of the peak frame (352 240) in this sequence is illustrated in Figure 3.5.

Figure 3.5: Frame from the peak at frame number 352 240 in the Mahalanobis distance scatter plot.

In this figure, the test subject seems to be looking out through the passenger seat window or in the rear mirror. Considering this was taken from the mostly normal behavior data, it was theorized that this indicated the model would potentially be able to catch more well defined anomalies as well.

In order to expand the Mahalanobis distance models, a temporal aspect was also implemented for the algorithm. This was done in a similar way as described for the kNN-method in Algorithm 1.

3.4 k-Nearest Neighbor

This section presents the methodology of the kNN method. How the data was pre-processed before usage, features used etc. The implementation of the method and how the evaluation was done is also described.

3.4.1 Pre-processing of data

Since the kNN- and the Mahalanobis methods are similar, the pre-processing was done in the same way. Features used for this method were head position, gaze direction, head rotation and mouth points. The dataset was cleaned from NaN-values, and two configurations were made, one with all data included, and one with quality corresponding to less than $\beta = 0.1$ filtered out.

3.4.2 Implementation

The implementation was done in two parts for the kNN method. The first part was the training, which calculates the norm $L_m[K]$ and the second part is the validation part.

In the training part, the pre-processed data was loaded and was divided into two sets, X_{N_1} and X_{N_2} with a ratio of 30/70%. By using Equation (2.6), a value for $L_K = L_m(K)$ was found for different feature configurations that were used. This was initially done with the parameter choices being $k = 1$, $s = 1$, $\gamma = 1$ and $\alpha = 0.05$ according to the implementation in paper [16]. However, a number of models were prepared by calculating different values of L_K for different configurations of parameter values. The different combinations of the parameters are shown in Table 3.4.

Table 3.4: The different combinations of parameter values, with all features in Table 3.2 used.

Exp.	k	s	α	γ	h	Train data	Validation data	L_K
1	1	1	0.05	1	2	filtered	filtered	0.3419009164794
2	1	1	0.05	1	2	unfiltered	filtered	0.2748069327814
3	1	1	0.05	1	2	filtered	unfiltered	0.3419009164794
4	5	5	0.05	1	2	unfiltered	unfiltered	1.4996053611797
5	10	10	0.05	1	2	unfiltered	unfiltered	3.1294382834934
6	10	5	0.05	1	2	unfiltered	unfiltered	3.1457832894561
7	10	2	0.05	1	2	unfiltered	unfiltered	3.0875982415333

In the next step, the different models were tested on a subset of the obtained normal behavior data, which the model had not been trained on to obtain the different values of L_K. As described in 2.1.2 k-Nearest Neighbor method, to use the model online to discover anomalies, the set X_{N_2} is kept as it was during training, while the new data being examined essentially acts as the new X_{N_1} set. For every new frame in X_{N_1} the distance metric L_t is computed and compared to L_K according to Equation (2.7). As also described in 2.1.2 k-Nearest Neighbor method, if D_t ends up being a positive value, that particular frame increases the accumulated anomaly score described in Algorithm 1.

By looking to the corresponding frames obtained as anomalies from the kNN-method with some combinations of the parameters noted in Table 3.4, some false alarms has been obtained which means that some of the frames had a normal behavior, note that the experiment was done on a validation-set of normal data. This means that if the driver does not look on the road, it can be seen as a anomaly. By looking at the corresponding frames the parameters was chosen to $k = 5$ and $s = 5$ on unfiltered data on both training and validation. After the parameters was found, the

training part was done on a big amount of data (50 000 data points with different drivers) on all feature combinations stated in Table 3.2. This data set are also used for the training part for the Mahalanobis distance method. Since the data set is divided 30/70 between X_{N_1} and X_{N_2}, it will take a lot of processor capacity to do all the norm-calculations in the validation part. To solve this problem, randomly selected points from X_{N_2} will be picked under the validation of the collected data. The optimal distance metric $L_m[K]$ and the parameters used from the training can be seen in Table 3.5.

3.4.3 Threshold tuning

The threshold tuning was done using a subset of the available dataset that contains mostly normal behavior data, which had not been used for training the models, similar to the approach for the Mahalanobis distance. Scatter plots and frames were then investigated to determine the performance of the models.

The subset used as X_{N_1} was created with the same data used for the performance test of the Mahalanobis method. This dataset was used for all parameter configurations in Table 3.4. For the cases with $k = 1$ and $s = 1$, the models didn't manage to find any frames contributing with a positive anomaly score, however with $k = 5$, $s = 5$ and $k = 10$, $s = 10$, both configurations managed to find an anomalous peak at the end of the interval, close to the interval where the Mahalanobis method managed to find an anomaly in Figure 3.3. This is displayed in Figure 3.6, for parameters $k = 5$, $s = 5$ trained on data not filtered by quality, and in Figure 3.7 for data where filtering was done.

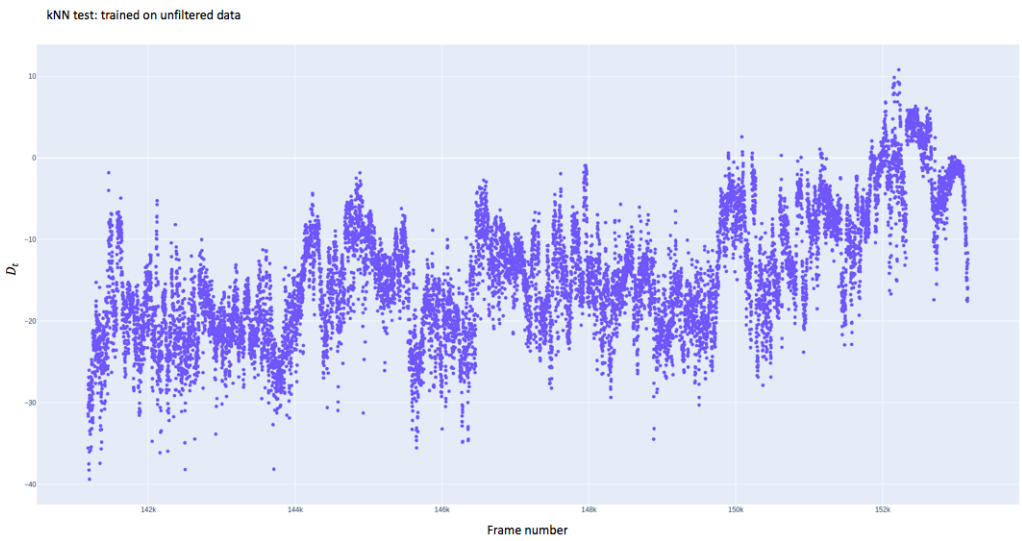

Figure 3.6: Scatter plot with the kNN-method with $k = 5$ and $s = 5$. Training based on unfiltered data.

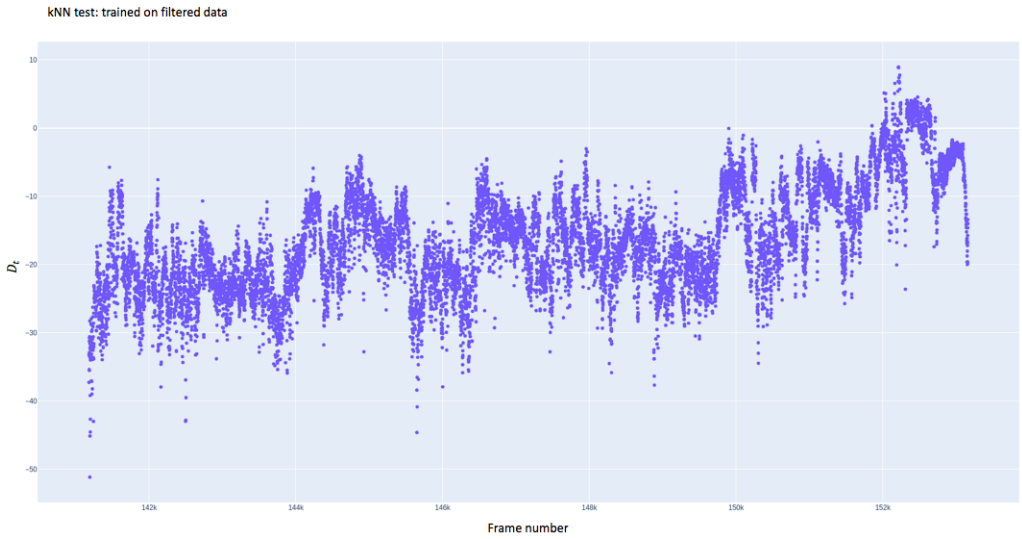

Figure 3.7: Scatter plot with the kNN-method with $k = 5$ and $s = 5$. Training based on filtered data.

As can be seen, the plots are fairly similar, and due to this the model obtained from using all data was used, as it was theorized to be more robust when applied to sequences with large amounts of bad quality data. By investigating what happens in the interval between frames $152\,476 - 152\,483$, where the peak is the highest, the frame displayed in Figure 3.8 was extracted from the video.

Figure 3.8: Frame of an anomaly obtained from the kNN-method on mostly normal driver behavior data.

Here the test subject seems to be looking down towards the stereo region of the cars interior, which can be considered a valid anomaly to find in the dataset being investigated, since it contains mostly normal driving behavior. Based on this, the parameter configuration of model 4 in Table 3.4 was considered to perform the best, since the configurations with larger values for k and s performed equally good, but needed more CPU-capacity to execute.

Finally, models with different features were trained to find their corresponding values for L_K, which are displayed in Table 3.5.

Table 3.5: Parameters for different feature combinations.

Features	k	s	α	γ	L_k
Head position, head rotation, and gaze direction	5	5	0.05	1	1.4996053611796876
Head position, head rotation, and mouth points	5	5	0.05	1	1.5678432765900523
Head rotation	5	5	0.05	1	0.6699943324781714
Head position	5	5	0.05	1	0.2189067041388315
Gaze direction	5	5	0.05	1	0.4350586555993561

3.5 Transformer

In this section, the implementation of the transformer model is described in detail. Data pre-processing and architectural decisions are explained, as well as the evaluation regarding the different trained models.

3.5.1 Pre-processing of data

For the transformer model, the data was pre-processed differently for different trained models. In other words, multiple models were trained with different configurations regarding the pre-processing of the data. For all models, the data points including missing values due to bad tracking were removed. The main difference between the models was that for some models, all available data was included, while for some models the data points including quality values $\beta < 0.1$ for some feature were removed.

Unlike the previously described statistical models, this was not so trivial as to just remove these data points and then start the training, due to the temporal aspect that the transformer takes into consideration. Each model was trained to regard a set time interval of length T, where each of these time intervals need to include data from consecutive frames, in order for the model to learn the correct patterns during the training. This meant that after the data points which included bad quality readings were removed, the remaining data needed to be manipulated such that each training sample used during training did not include a gap in its sequence, due to a bad quality value being removed.

3.5.2 Implementation

The input to the model consists of the features, being regarded as a multivariate sequence of the T previous frames. The 1D-convolution layer then performs convolutions over all features in every time step separately with dim number of filters, which essentially creates the input embedding to the self attention module of size $T \times dim$. This is then fed to the attention module, which consists of N identical sequentially stacked encoders. The output from this attention module is the attention score between the different time steps, and has the same dimension $T \times dim$ as the input to the attention module. To ensure that the attention is applied in a way which maintains temporal causality, a so called masked self attention is applied. This means that the attention mechanism is masked, such that for a given time step τ, the attention module only regards inputs from earlier time steps $\tau - i$ in the sequence, where $i = 1, 2, .., \tau - 1$. Finally, the output from the attention module is sent through a number of fully connected layers, which reduces the inputs dimension down to the number of features being regarded, and produces the prediction for the values of the features in the next time step.

The transformer model was implemented to take the features from the previous T frames in the sequence, and then try to predict the values of the features in the next time step $t + 1$, where t is the last time step which has been fed to the model.

To get an idea of what hyper parameters and architectural structure of the network seemed most optimal, as well as what type of data pre-processing was most successful, a number of different models were trained on a subset of the available data. All of these models were trained on the features head position, head rotation and gaze direction listed in Table 3.2. In total, eight different model configurations were trained, which are displayed in Table 3.6.

Table 3.6: Transformer model architectures.

Model number	dim/ heads	Normalization	Quality threshold	MSE
1	64/4	-	-	0.938107096354
2	64/4	std	-	0.794613931174
3	64/4	-	0.1	0.476186639367
4	64/4	std	0.1	0.584814380157
5	128/8	-	-	0.003690199536
6	128/8	std	-	0.938107096354
7	128/8	-	0.1	0.003313821575
8	128/8	std	0.1	0.654403046332

The dim/heads column describes the size of the embedding dimension and the number of heads in the encoder blocks. The pre-processing consisted either of leaving the data x as it was or using standardization to obtain $\bar{x}$ for every feature separately, i.e. subtracting the mean μ and scaling with the standard deviation σ for every data point, according to

$$\bar{x} = \frac{x - \mu}{\sigma} \tag{3.3}$$

Finally, the quality threshold describes the threshold for which the data with a quality less than that threshold was removed.

The models were trained over 5 epochs with Mean Squared Error (MSE) as loss function. After this they were all tested on the same validation set, where each model attempted to reconstruct the signal of one of the features over an interval of 3000 frames. The plots from the two best performing models based on MSE value is displayed in Figure 3.9 and 3.10.

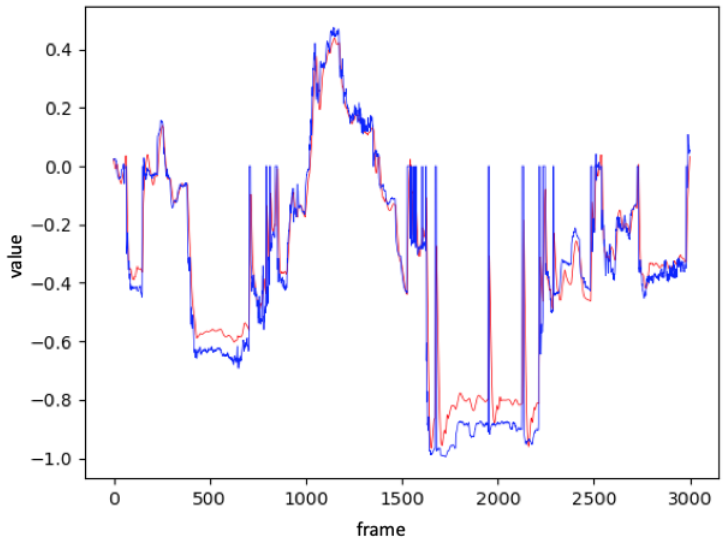

Figure 3.9: Plot of predicted (red) and ground truth (blue) of the feature "Head Position X" using model 5 from Table 3.6.

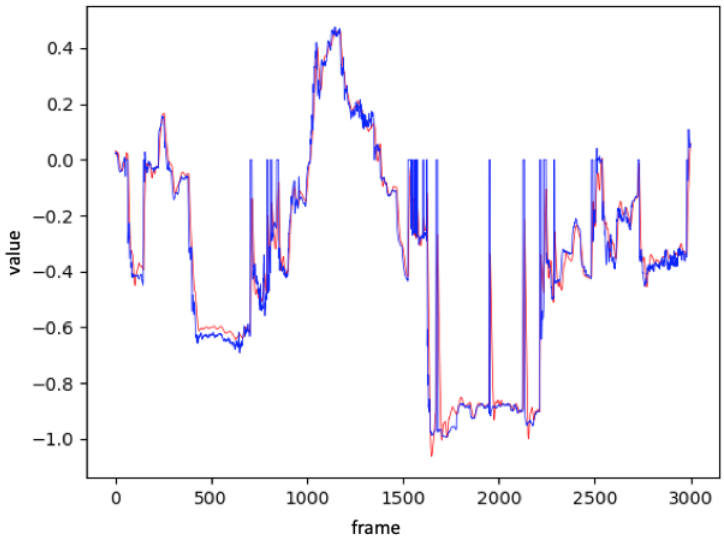

Figure 3.10: Plot of predicted (red) and ground truth (blue) of the feature "Head Position X" using model 7 from Table 3.6.

Based on the MSE value and from examining the plots, model 5 and 7 were considered to be performing best, which ended up being the architectures to examine more thoroughly.

3.5.3 Performance test

The next step was to choose which features were to be included in the transformer model. To do this, the first step was to train two models, with architectures according to model 5 and 7, on all available data. For these models, the time frame being considered was chosen as $T = 120$. These models were trained over 10 epochs, and then the models were applied to the same subset of mostly normal data (which was not used during training of the models) used to determine the performance of the statistical models. Just like for the statistical models, the results were plotted, and these are shown in Figure 3.11 and 3.12.

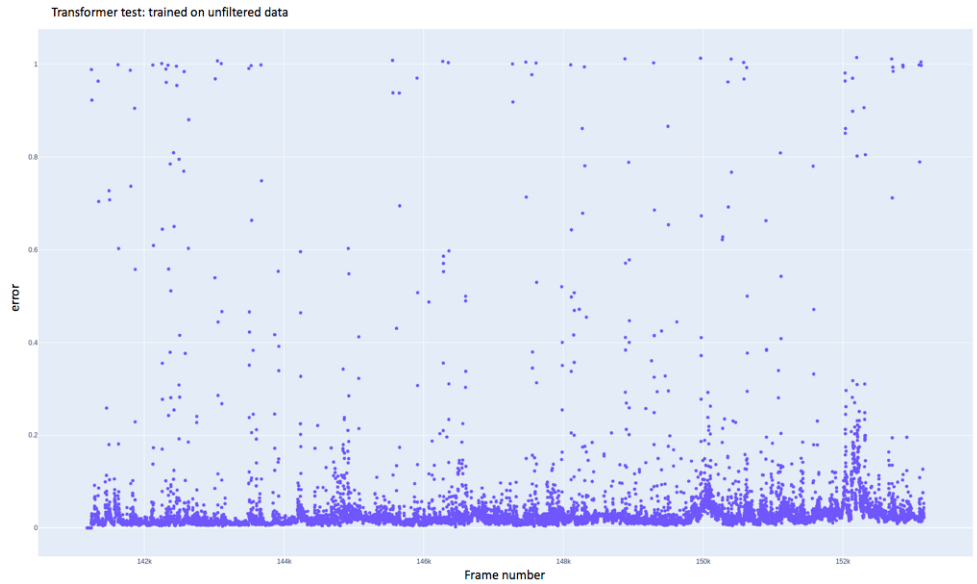

Figure 3.11: Plot of reconstruction error of Transformer network trained on all data, applied to the validation subset of the mostly normal data.

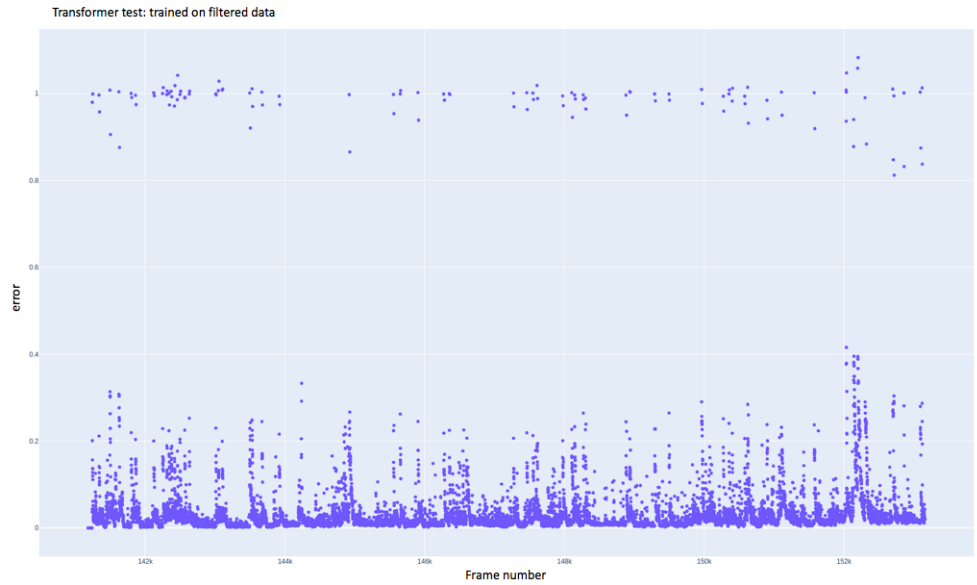

Figure 3.12: Plot of reconstruction error of Transformer network trained on all data with quality $\beta > 0.1$, applied to the validation subset of the mostly normal data.

From these plots it was obvious that neither of these iterations were good enough to apply to the problem. From examining the data, and looking at results from the statistical methods, it was theorized that the reason for this poor result was due to the transformer being very sensitive to bad quality data when trying to predict the next time step. The bad quality data mainly comes from the gaze direction feature. This also has the additional negative effect on the model trained on data with quality $\beta > 0.1$, where a lot of training samples get removed due to the need of

continuous intervals during training. It was therefore decided that the gaze direction feature should be omitted, and instead the mouth points features from Table 3.2 was added. A new model using the features head position, head rotation and mouth points were trained over 10 epochs. This new model was then applied to the same subset of the mostly normal data as the other methods. The reconstruction error of this sequence is displayed in Figure 3.13.

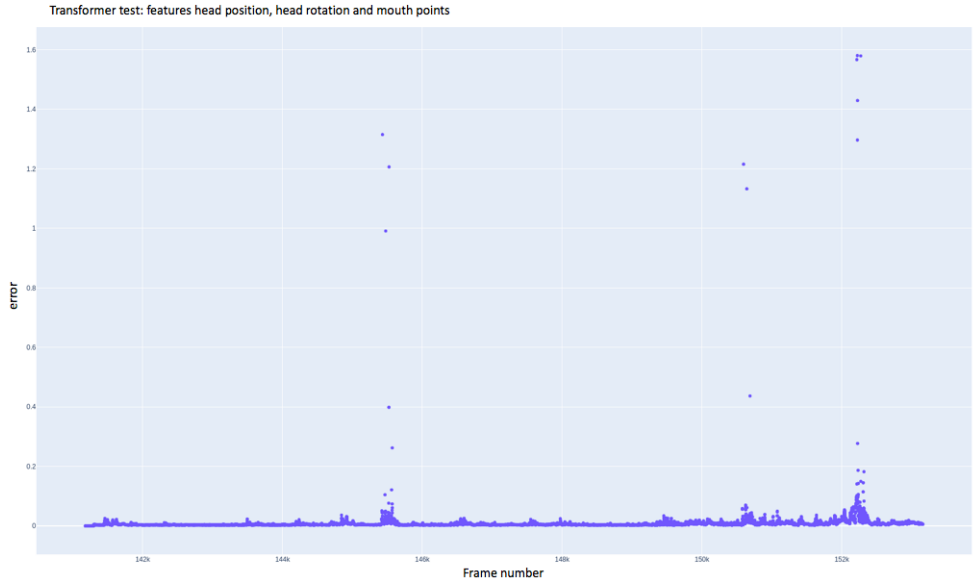

Figure 3.13: Plot of reconstruction error of Transformer network using the features head position, head rotation and mouth points.

As can be seen in Figure 3.13, the model seems a lot more stable after dropping the gaze direction feature. There are three clear passages in the plot where the reconstruction error gets especially high over a number of frames. To begin with, a clear peak is found at the end of the interval. By extracting the frames from the peak, it is found that these frames belong to the same interval as was found using the Mahalanobis distance method, illustrated in Figure 3.5. There are also two additional peaks present in the graph however, which are investigated in a similar way, and the frames belonging to the peak points in these two intervals are illustrated in Figure 3.14 and 3.15.

Figure 3.14: Frame belonging to the first peak in Figure 3.13.

Figure 3.15: Frame belonging to the second peak in Figure 3.13.

From the frames it can be seen that in both cases, the driver is covering his mouth, causing the tracking of the mouth feature to be lost. This is the reason for the reconstruction error being so high, and although these events can not be considered anomalies in the sense defined in this project, the model is still deemed to perform very well based on the results in Figure 3.13. If the passages connected to the high peaks in the graph are omitted, the rest of the reconstruction error is very small. By looking closer at the rest of the graph, the value $\epsilon = 0.03$ was considered a good

threshold, as this ensured that no other peaks than the three obvious ones broke this threshold.

3.6 Evaluation

To get a fair comparison between the statistical methods and the transformer, the percentage of the amount of frames obtained at each scenario are calculated. The percentage of normal frames of the normal driving scenarios are also calculated and compared for each model. Finally a total accuracy will be calculated for both normal and anomalous driving where the individual accuracy for each scenario will be added together and are divided with the total number of scenarios.

First the accuracy for the anomaly frames will be calculated for each scenario. This was done using Equation (3.4).

$$a_{anomaly}^{scenario\ x} = \frac{N_{tot}^{scenario\ x} - N_{normal}^{scenario\ x}}{N_{tot}^{scenario\ x}} \tag{3.4}$$

where $N_{tot}^{scenario\ x}$ is the total frames in anomaly scenario x and $N_{normal}^{scenario\ x}$ is the total amount of frames that the model detect as normal. In the same way, the accuracy of the normal driver behavior is calculated and can be seen in Equation (3.5).

$$a_{normal}^{scenario\ y} = \frac{N_{tot}^{scenario\ y} - N_{anomaly}^{scenario\ y}}{N_{tot}^{scenario\ y}} \tag{3.5}$$

where $N_{tot}^{scenario\ y}$ is the total frames in the normal scenario y and $N_{anomaly}^{scenario\ y}$ is the total amount of anomaly frames detected from the model within this scenario. The total accuracy of the anomaly detection for each model is then calculated as

$$a_{anomaly,\ tot} = \frac{1}{N_{tot}^{a}} \sum_{k=1}^{N_a} N_{anomaly}^{scenario\ k} \tag{3.6}$$

where N_{tot}^{a} is the total amount of frames classified as anomalies. $k = 1,\ ...\ N_a$ where N_a is the total anomaly scenarios and $N_{anomaly}^{scenario\ k}$ is the total amount of frames that the model obtained as anomalies for each anomaly scenario k (11 in total). To calculate the total accuracy of the normal driver behavior for each model, Equation (3.7) is used.

$$a_{normal,\ tot} = \frac{1}{N_{tot}^{n}} \sum_{k=1}^{N_n} N_{normal}^{scenario\ k} \tag{3.7}$$

where N_{tot}^{n} is the total amount of frames classified as normal driver behavior. $k = 1,\ ...\ , N_n$ where N_n is the total amount of normal scenarios and $N_{normal}^{scenario\ k}$ is the total amount of frames that the model obtained as normal for each normal scenario k (11 in total).

4

Results

In this section the results are present. It includes the accuracy to find anomalies for each method and the corresponding features used for every model. The total amount of anomalies obtained in the anomaly scenarios and the total amount of normal frames are also presented. A time-line on how the validation set is structured can be seen in Figure 3.1.

4.1 Mahalanobis Distance

Here the results from the statistical method Mahalanobis distance are stated. It includes different model configurations based on the head position, head rotation, gaze direction and mouth points. One plot shows the detected anomalies and a table is stated with the accuracy for each model, averaged over all the validation IDs. The parameters used for the different configurations are also stated.

4.1.1 Head position

When only looking at head position, the threshold was set to $\epsilon = 3$ for both of the Mahalanobis distance methods (with and without temporal aspect). A plot of the obtained anomalies can be seen in Figure 4.1.

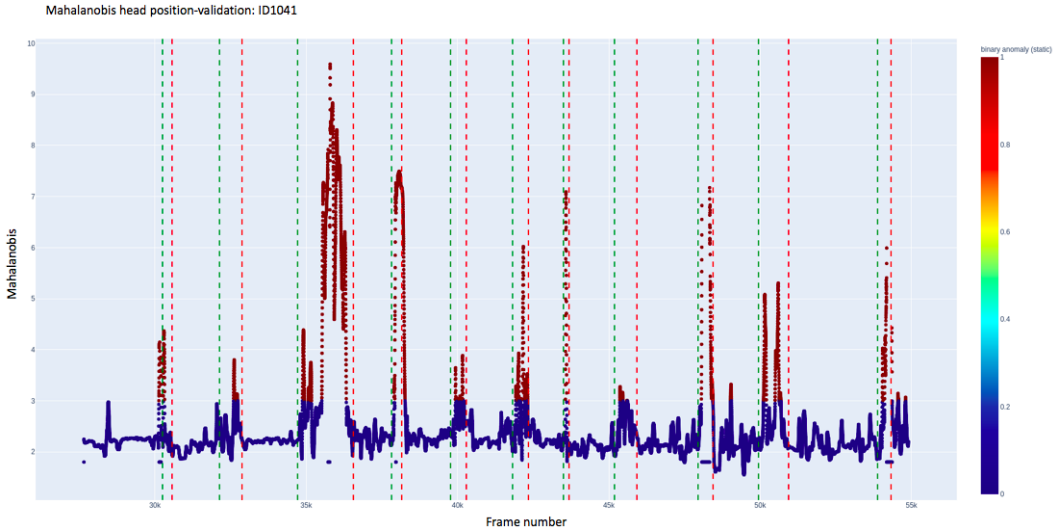

Figure 4.1: Plot of the Mahalanobis distance on the validation set of ID1041 with head position as feature, without temporal aspect.

The green lines in Figure 4.1 denote the start of each anomaly scenario, the red lines denote the stop of each scenario, the blue points are points detected as normal and the brown points are the points detected as anomalies. The total amount of anomalies, normal data and the accuracy's can be seen in Table 4.1.

Table 4.1: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head position, without the temporal aspect.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.383730	0.989322	3269	8519	17141	17326
2	0.026141	0.990778	256	9793	10744	10844
3	0.447991	0.991921	7438	16603	11295	11387
4	0.040493	0.993944	243	6001	12309	12384
5	0.211248	0.983031	1927	9122	14425	14674
6	0.197376	0.972511	1700	8613	13267	13642
7	0.068876	0.994705	212	3078	10521	10577
8	0.073792	0.905326	687	9310	10997	12147
9	0.181735	0.976198	1190	6548	13042	13360
10	0.152899	0.970501	1292	8450	9541	9831
11	0.303716	0.961428	1463	4817	12388	12885
Total:	**0.216578**	**0.975643**	**19677**	**90854**	**135670**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

The Mahalanobis distance model with the temporal aspect had a threshold $h = 10$. An example of the detected anomalies can be seen in Figure 4.2.

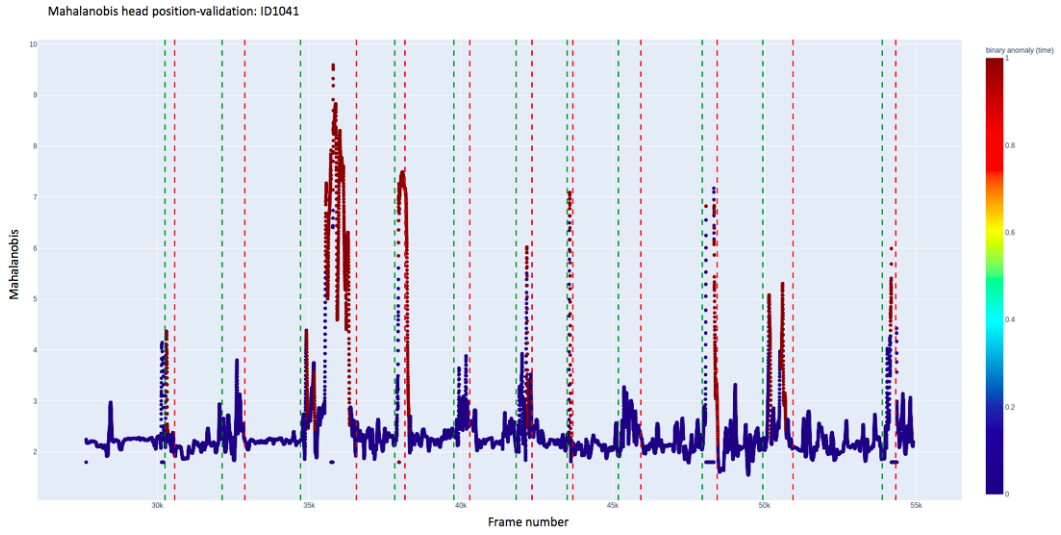

Figure 4.2: Plot of the Mahalanobis distance on the validation set of ID1041 with head position as feature, with temporal aspect.

The total amount of frames detected as anomalies- and normal data and the corresponding accuracy can be seen in Table 4.2.

Table 4.2: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head position, with the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.363892	0.994921	3100	8519	17238	17326
2	0.000000	0.998986	0	9793	10833	10844
3	0.421731	0.991482	7002	16603	11290	11387
4	0.035994	0.997335	216	6001	12351	12384
5	0.175510	0.988824	1601	9122	14510	14674
6	0.167537	0.970825	1443	8613	13244	13642
7	0.033138	0.991869	102	3078	10491	10577
8	0.022771	0.936198	212	9310	11372	12147
9	0.129047	0.989296	845	6548	13217	13360
10	0.089467	0.974469	756	8450	9580	9831
11	0.246834	0.962825	1189	4817	12406	12885
Total:	**0.181235**	**0.981841**	**16466**	**90854**	**136532**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.1.2 Gaze direction

This Mahalanobis model used only the gaze direction as feature. The threshold was set to $\epsilon = 2.4$ for both of the configurations. In Figure 4.3 a plot of the detected anomalies for the model without temporal aspect can be seen.

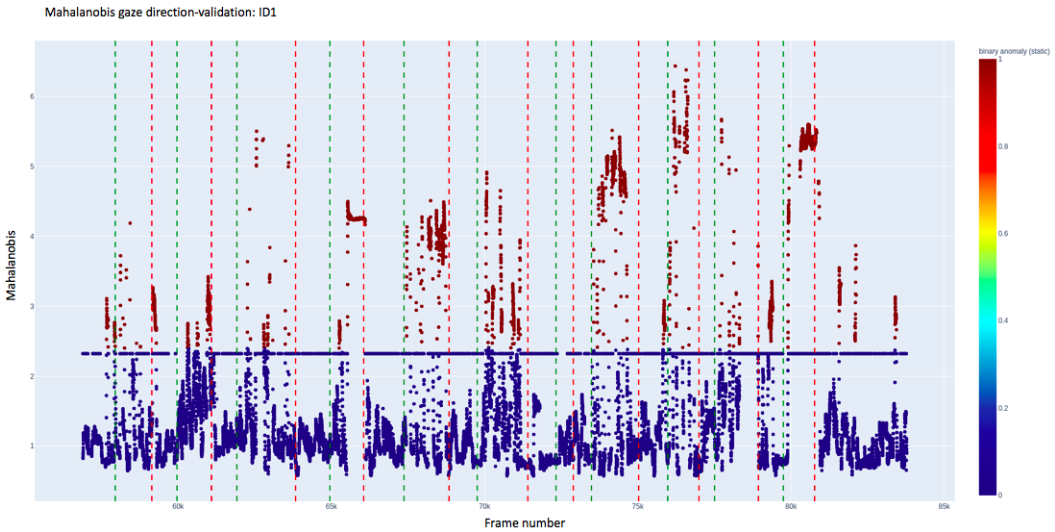

Figure 4.3: Plot of the Mahalanobis distance on the validation set of ID1 with gaze direction as feature, without temporal aspect.

A "belt" of bad quality points are visualized at approximately $D = 2.3$. The total amount of anomalies and normal data, and the accuracy for each scenario can be seen in Table 4.3.

Table 4.3: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using gaze direction, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.056814	0.850456	484	8519	14735	17326
2	0.169713	0.882792	1662	9793	9573	10844
3	0.059266	0.914288	984	16603	10411	11387
4	0.227962	0.932897	1368	6001	11553	12384
5	0.403859	0.843192	3684	9122	12373	14674
6	0.217230	0.899575	1871	8613	12272	13642
7	0.003899	0.906401	12	3078	9587	10577
8	0.314071	0.874949	2924	9310	10628	12147
9	0.082926	0.809356	543	6548	10813	13360
10	0.067456	0.828197	570	8450	8142	9831
11	0.157359	0.877532	758	4817	11307	12885
Total:	**0.163559**	**0.872980**	**14860**	**90854**	**121394**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

For the Mahalanobis distance model with the temporal aspect included, a threshold $h = 10$ was set. An example of the detected anomalies on driver ID1 can be seen in Figure 4.4.

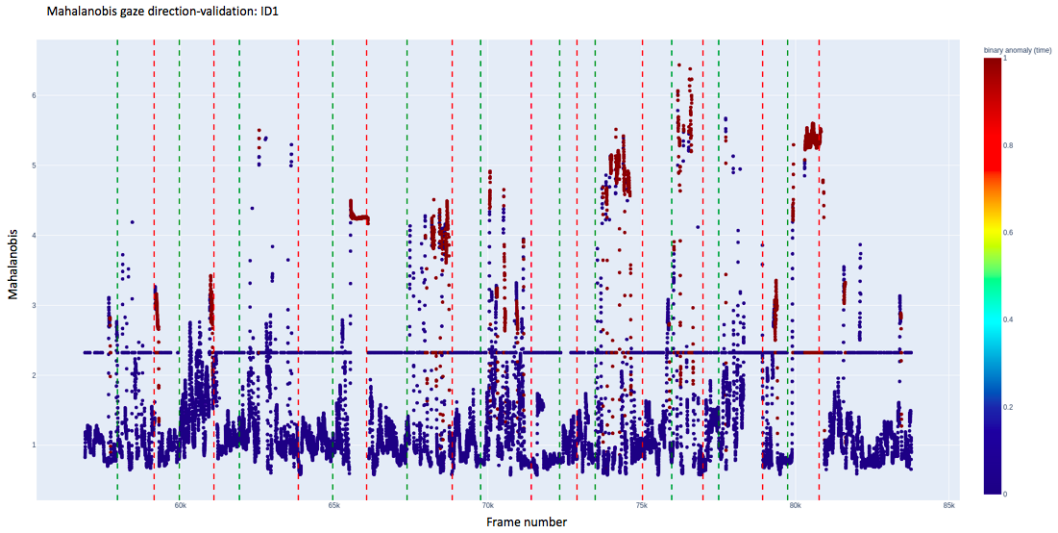

Figure 4.4: Plot of the Mahalanobis distance on the validation set of ID1 with gaze direction as feature, with temporal aspect included.

By looking at Figure 4.4, the belt of bad quality is the same as for the Mahalanobis distance model without the temporal aspect. Clear spikes are also obtained during the different anomaly scenarios. The total amount of anomalous and normal frames, and the accuracy obtained for the model under the different scenarios can be seen in Table 4.4.

Table 4.4: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using gaze direction, with the temporal aspect included.

Scenario	**A. acc.**	**N. acc.**	**A. f.**	**Tot. f. A.**	**N. f.**	**Tot. f. N.**
1	0.101303	0.901824	863	8519	15625	17326
2	0.144695	0.915714	1417	9793	9930	10844
3	0.059086	0.953544	981	16603	10858	11387
4	0.191968	0.961160	1152	6001	11903	12384
5	0.517759	0.854845	4723	9122	12544	14674
6	0.127714	0.942311	1100	8613	12855	13642
7	0.000000	0.936466	0	3078	9905	10577
8	0.342535	0.901210	3189	9310	10947	12147
9	0.154856	0.813698	1014	6548	10871	13360
10	0.074438	0.890652	629	8450	8756	9831
11	0.190783	0.919441	919	4817	11847	12885
Total:	**0.175963**	**0.906398**	**15987**	**90854**	**126041**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.1.3 Head rotation

The threshold for the Mahalanobis distance method with and without the temporal aspect is set to $\epsilon = 4$. These models has only used the feature head rotation. A plot of the Mahalanobis distance without the temporal aspect can be seen in Figure 4.5 on ID1.

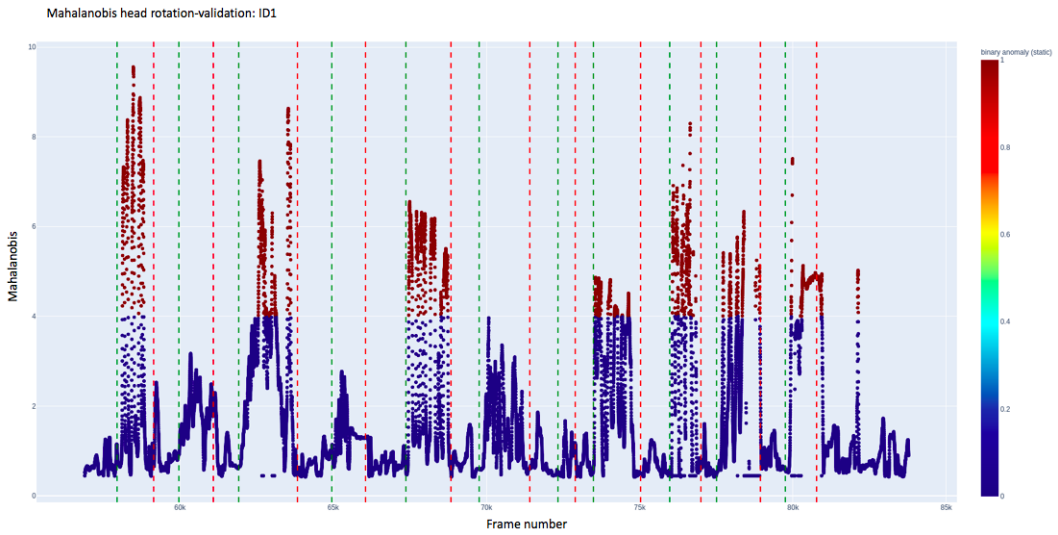

Figure 4.5: Plot of the Mahalanobis distance on the validation set of ID1 with head rotation as feature, without temporal aspect included.

By looking at Figure 4.5, anomaly scenario 2, 4, 6 and 7 does not detect any anomalies for this model configuration. The total amount of anomaly frames, normal frames and the accuracy for each scenario can be seen in Table 4.5.

Table 4.5: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head rotation, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.340533	0.994286	2901	8519	17227	17326
2	0.007352	0.995850	72	9793	10799	10844
3	0.114437	0.998595	1900	16603	11371	11387
4	0.000000	0.998595	0	6001	12362	12384
5	0.445626	0.983917	4065	9122	14438	14674
6	0.023801	0.994136	205	8613	13562	13642
7	0.000000	0.998224	0	3078	10523	10577
8	0.051880	0.983453	483	9310	11946	12147
9	0.110263	0.970359	722	6548	12964	13360
10	0.095030	0.999593	803	8450	9827	9831
11	0.119369	0.969810	575	4817	12496	12885
Total:	**0.129064**	**0.988911**	**11726**	**90854**	**137515**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

For the Mahalanobis model with temporal aspect for the head rotation as feature, a threshold was set to $h = 20$. A plot of the Mahalanobis distance computed for this model configuration on ID1 can be seen in Figure 4.6.

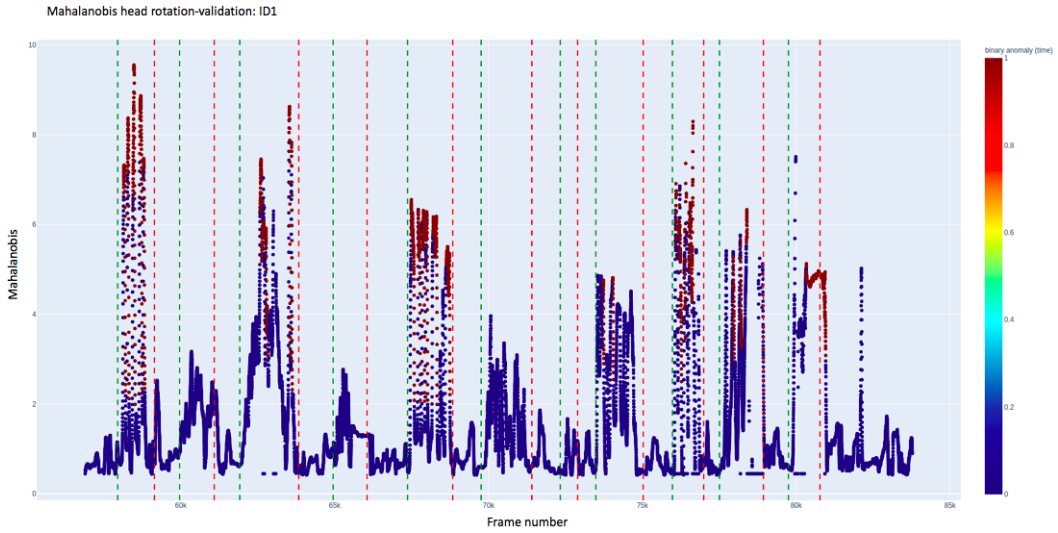

Figure 4.6: Plot of the Mahalanobis distance on the validation set of ID1 with head rotation as feature, with temporal aspect included.

In Figure 4.6 clear spikes can be seen for some anomalous scenarios, especially scenario 1 and 3 which corresponds to look behind the shoulder and taking off the jacket. The total anomaly- and normal frames for this model configuration, including the accuracy can be seen in Table 4.6.

Table 4.6: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head rotation, with the temporal aspect included.

Scenario	**A. acc.**	**N. acc.**	**A. f.**	**Tot. f. A.**	**N. f.**	**Tot. f. N.**
1	0.309778	0.996364	2639	8519	17263	17326
2	0.000000	0.996311	0	9793	10804	10844
3	0.058062	0.997102	964	16603	11354	11387
4	0.000000	0.993379	0	6001	12302	12384
5	0.358474	0.984735	3270	9122	14450	14674
6	0.009753	0.993696	84	8613	13556	13642
7	0.000000	0.993389	0	3078	10507	10577
8	0.005048	0.993002	47	9310	12062	12147
9	0.063531	0.998054	416	6548	13334	13360
10	0.058225	0.998169	492	8450	9813	9831
11	0.091758	0.978813	442	4817	12612	12885
Total:	**0.091949**	**0.992809**	**8354**	**90854**	**138057**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.1.4 Head position, head rotation and gaze direction

Here, multiple features are combined. In this case the Mahalanobis distance method is computed using the features head position, head rotation and gaze direction. The threshold for the model with and without the temporal aspect is $\epsilon = 5$. An example of the detected anomalies without the temporal aspect can be seen in Figure 4.7.

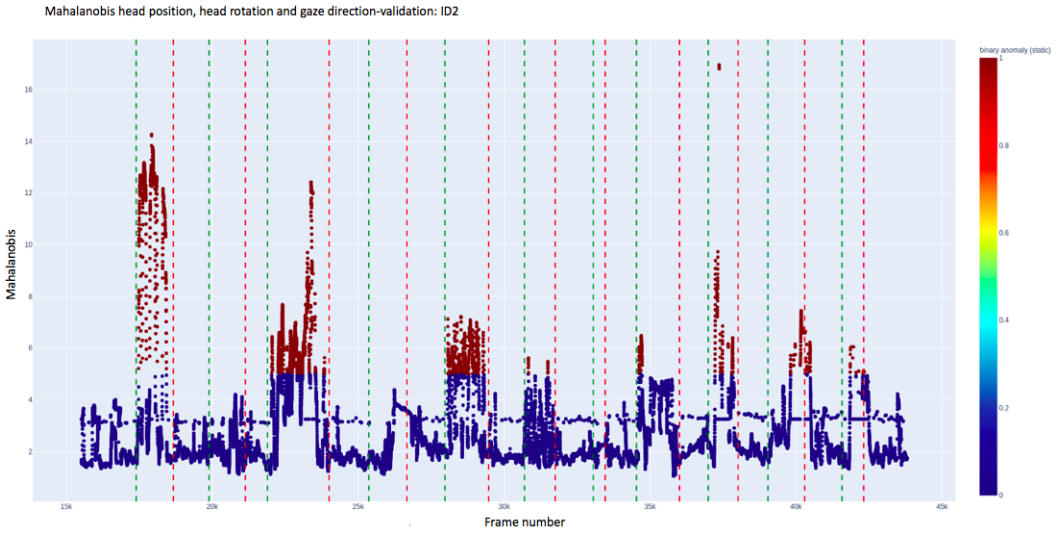

Figure 4.7: Plot of the Mahalanobis distance on the validation set of ID2 with head position, head rotation and gaze direction as features, without temporal aspect included.

By looking in Figure 4.7, a "belt" of points can be seen in this plot as well. This corresponds to the bad-quality data from the gaze direction points, but has less effect on the detection of anomalies for this configuration. The accuracy, and the amount of frames obtained as normal and anomalies can be seen in Table 4.7.

Table 4.7: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head position, head rotation and gaze direction, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.406151	0.988688	3460	8519	17130	17326
2	0.011641	0.993729	114	9793	10776	10844
3	0.444920	0.995170	7387	16603	11332	11387
4	0.045159	0.992894	271	6001	12296	12384
5	0.486406	0.971446	4437	9122	14255	14674
6	0.078138	0.981821	673	8613	13394	13642
7	0.009422	1.000000	29	3078	10577	10577
8	0.229431	0.944595	2136	9310	11474	12147
9	0.159133	0.937201	1042	6548	12521	13360
10	0.144260	0.988099	1219	8450	9714	9831
11	0.225452	0.956151	1086	4817	12320	12885
Total:	**0.240539**	**0.976498**	**21854**	**90854**	**135789**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

The Mahalanobis distance model with temporal aspect has a threshold $h = 20$. A scatter plot on the validation set of driver ID2 can be seen in Figure 4.8.

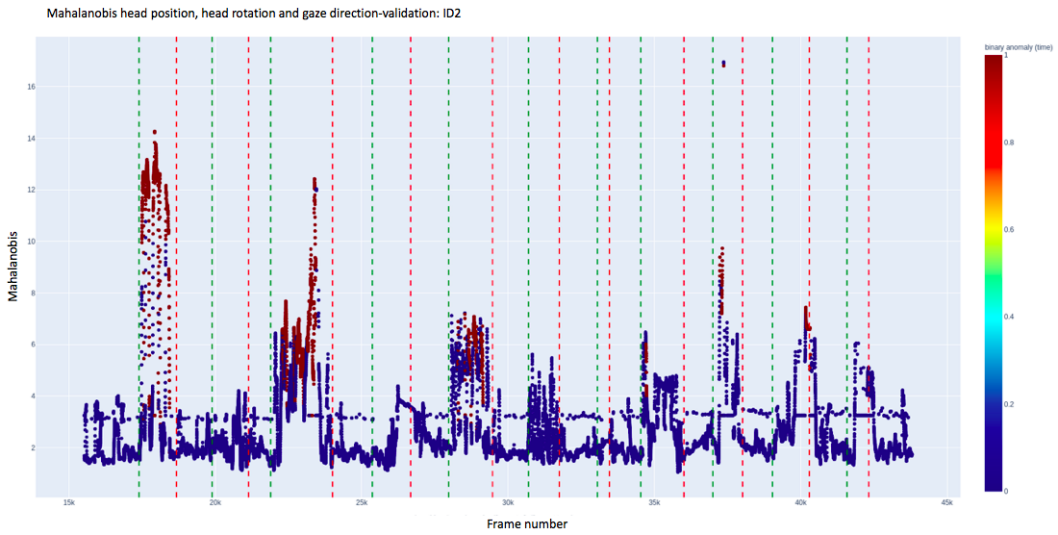

Figure 4.8: Plot of the Mahalanobis distance on the validation set of ID2 with head position, head rotation and gaze direction as features, with temporal aspect included.

The anomalous scenarios 2, 4, 6, 7 and 11 are hard to detect. This corresponds to drunk, heart attack, panic-attack, shocked and sleep. Some points are also obtained in scenario 9 with a high Mahalanobis distance value in relation to the rest of the points. The accuracy for this model configuration can be seen in Table 4.8.

Table 4.8: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head position, head rotation and gaze direction, with the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.398756	0.996248	3397	8519	17261	17326
2	0.002553	0.996127	25	9793	10802	10844
3	0.415226	0.992535	6894	16603	11302	11387
4	0.034161	0.989745	205	6001	12257	12384
5	0.407915	0.974308	3721	9122	14297	14674
6	0.046558	0.985486	401	8613	13444	13642
7	0.007472	0.994895	23	3078	10523	10577
8	0.114715	0.980654	1068	9310	11912	12147
9	0.103390	0.971332	677	6548	12977	13360
10	0.062840	0.986573	531	8450	9699	9831
11	0.161511	0.956539	778	4817	12325	12885
Total:	**0.195038**	**0.983762**	**17720**	**90854**	**136799**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.1.5 Head position, head rotation and mouth points

This Mahalanobis distance method combination is using the features head position, head rotation and mouth points. The threshold was set to $\epsilon = 6$. The model was created both with and without the temporal aspect. First the results for the Mahalanobis distance model without temporal aspect is shown.

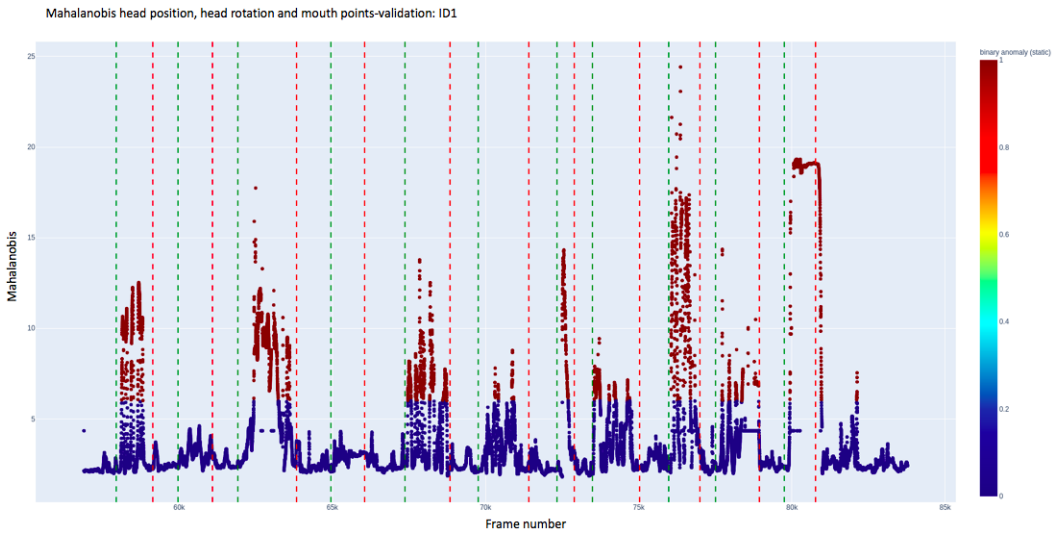

Figure 4.9: Plot of the Mahalanobis distance on the validation set of ID1 with head position, head rotation and mouth points as features, without temporal aspect included.

As can be seen in Figure 4.9, some of the anomalous scenarios are hard to obtain. Especially the anomaly scenario 2, drunk. The overall anomaly- and normal accuracy, and the results for each scenario individually for the model configuration can be seen in Table 4.17.

Table 4.9: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head position, head rotation and mouth points, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.401690	0.984763	3422	8519	17062	17326
2	0.005208	0.985522	51	9793	10687	10844
3	0.413118	0.999034	6859	16603	11376	11387
4	0.116481	0.996689	699	6001	12343	12384
5	0.424249	0.965040	3870	9122	14161	14674
6	0.066643	0.969726	574	8613	13229	13642
7	0.179012	0.992531	551	3078	10498	10577
8	0.042213	0.969787	393	9310	11780	12147
9	0.161423	0.968039	1057	6548	12933	13360
10	0.115621	0.991049	977	8450	9743	9831
11	0.184555	0.968025	889	4817	12473	12885
Total:	**0.212891**	**0.980065**	**19342**	**90854**	**136285**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

The model configuration with the temporal aspect included has a threshold $h = 20$. An example of the obtained anomalies on driver ID1 in the validation set can be seen in Figure 4.9.

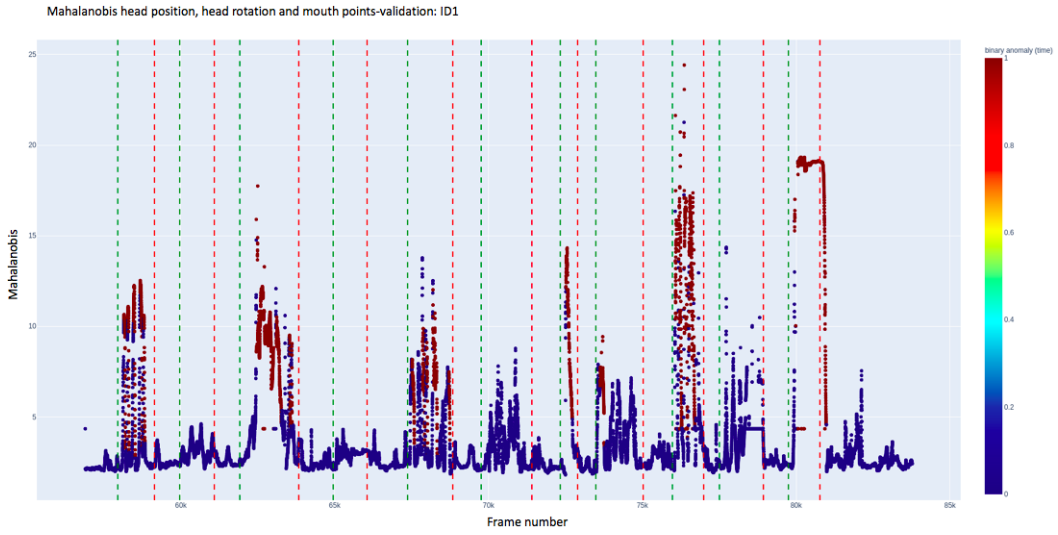

Figure 4.10: Plot of the Mahalanobis distance on the validation set of ID1 with head position, head rotation and mouth points as features, with temporal aspect included.

The total anomaly- and normal accuracy, as well as total number of frames identified in each category, along with each normal- and anomaly scenario for the model configuration can be seen in Table 4.10.

Table 4.10: Accuracy and total number of frames identified as anomalies and normal behavior for the Mahalanobis distance method using head position, head rotation and mouth points, with the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.412020	0.993536	3510	8519	17214	17326
2	0.000000	0.996865	0	9793	10810	10844
3	0.405348	0.994292	6730	16603	11322	11387
4	0.078154	0.988453	469	6001	12241	12384
5	0.342030	0.964699	3120	9122	14156	14674
6	0.048415	0.967820	417	8613	13203	13642
7	0.167641	0.982982	516	3078	10397	10577
8	0.014715	0.978596	137	9310	11887	12147
9	0.105223	0.986901	689	6548	13185	13360
10	0.054911	0.983928	464	8450	9673	9831
11	0.166494	0.974156	802	4817	12552	12885
Total:	**0.185506**	**0.982619**	**16854**	**90854**	**136640**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.2 k-Nearest Neighbor

The kNN method are also used on separately- and combined features. The total anomaly- and normal frames for each scenario are stated in tables and plots are visualized on the detection of the anomalies obtained for the different model combinations. Since the gaze direction points has a lot of bad quality data, the norm of L_k will end up in zero. Which results in a undefined value in equation 2.7. Therefore the feature with only gaze direction are not investigated. All the model configurations for the kNN methods used 2 000 random values in the subset X_{N_2}.

4.2.1 Head position

The kNN method using head position as feature is presented with and without the temporal aspect in this section. The threshold for both the model configurations is set to $L_K = 0.2189067041388315$. The computed kNN-distance for the model without the temporal aspect can be seen in Figure 4.11.

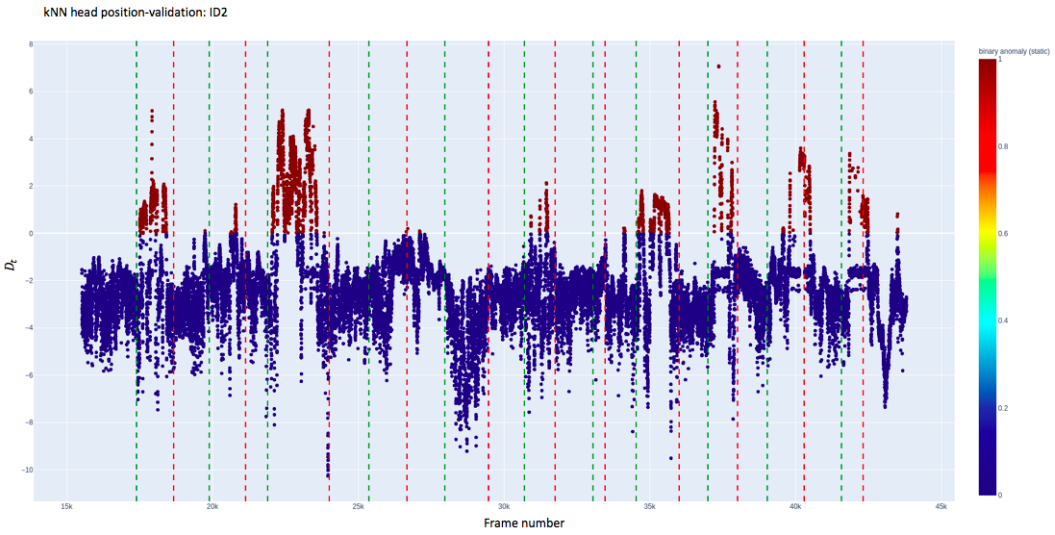

Figure 4.11: Plot of the kNN-distance on the validation set of ID2 with head position as feature, without temporal aspect included.

In Figure 4.11, very noisy values D_t are obtained. The model can not find anomalies at scenario 4, 5 and 7. The total accuracy for the anomalous- and normal scenarios, and the total amount of frames obtained for each category can be seen in Table 4.11.

Table 4.11: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head position, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.401103	0.919081	3417	8519	15924	17326
2	0.161850	0.958226	1585	9793	10391	10844
3	0.506957	0.930623	8417	16603	10597	11387
4	0.268622	0.899225	1612	6001	11136	12384
5	0.273405	0.881355	2494	9122	12933	14674
6	0.272495	0.847750	2347	8613	11565	13642
7	0.175114	0.876714	539	3078	9273	10577
8	0.291192	0.850910	2711	9310	10336	12147
9	0.232285	0.895808	1521	6548	11968	13360
10	0.316923	0.882413	2678	8450	8675	9831
11	0.362259	0.827939	1745	4817	10668	12885
Total:	**0.319919**	**0.887881**	**29066**	**90854**	**123466**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

The kNN method with feature head position with the temporal aspect included has a threshold $h = 10$. An example of the computed kNN-distance for driver ID2 of this model configuration can be seen in Figure 4.12.

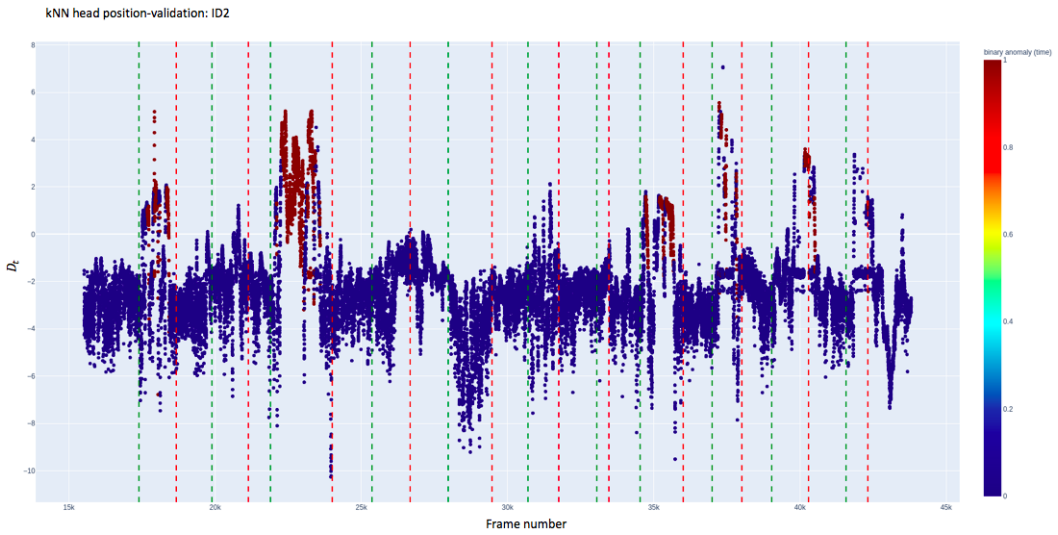

Figure 4.12: Plot of the kNN-distance on the validation set of ID2 with head position as feature, with temporal aspect included.

In Figure 4.12, the validation-set on ID2 was used. The kNN method with the temporal aspect can obtain the majority of anomalies in scenario 4 (taking off the jacket). The accuracy and the total frames obtained for each scenario can be seen in Table 4.12.

Table 4.12: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head position, with the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.325860	0.988572	2776	8519	17128	17326
2	0.100276	0.986629	982	9793	10699	10844
3	0.468349	0.966190	7776	16603	11002	11387
4	0.194968	0.978198	1170	6001	12114	12384
5	0.215852	0.940303	1969	9122	13798	14674
6	0.241612	0.891365	2081	8613	12160	13642
7	0.070175	0.868205	216	3078	9183	10577
8	0.206015	0.890673	1918	9310	10819	12147
9	0.143250	0.967141	938	6548	12921	13360
10	0.232781	0.933577	1967	8450	9178	9831
11	0.298941	0.877066	1440	4817	11301	12885
Total:	**0.255717**	**0.937047**	**23233**	**90854**	**130303**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.2.2 Head rotation

This kNN model is based on head rotation as feature, and was implemented with and without the temporal aspect included. The threshold for L_K was set to $L_K = 0.6699943324781714$. An example of the computed kNN-distance for person ID 56 can be seen in Figure 4.13.

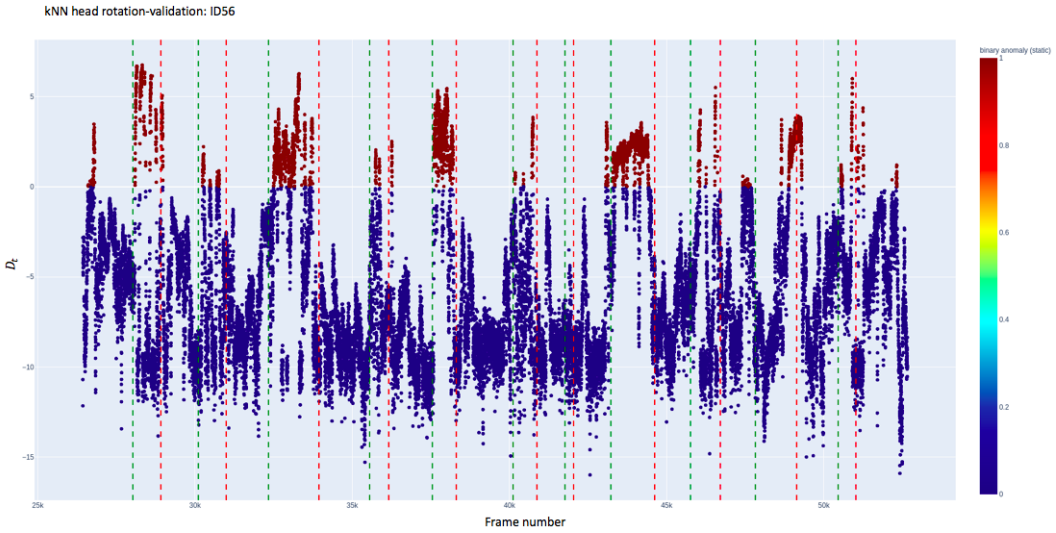

Figure 4.13: Plot of the kNN-distance on the validation set of ID56 with head rotation as feature, without temporal aspect included.

The kNN method without the temporal aspect manages to at least label some points in every anomalous scenario as anomalies. Some anomalies are also obtained in the beginning of the recording, which corresponds to only driving normal. The accuracy, amount of anomalies and normal data frames obtained on each scenario can be seen in Table 4.13.

Table 4.13: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head rotation, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.361897	0.988976	3083	8519	17135	17326
2	0.230879	0.970859	2261	9793	10528	10844
3	0.366018	0.972688	6077	16603	11076	11387
4	0.120647	0.974241	724	6001	12065	12384
5	0.443982	0.929399	4050	9122	13638	14674
6	0.113665	0.966134	979	8613	13180	13642
7	0.025341	0.975324	78	3078	10316	10577
8	0.498711	0.877501	4643	9310	10659	12147
9	0.204032	0.915793	1336	6548	12235	13360
10	0.203905	0.953616	1723	8450	9375	9831
11	0.193066	0.949166	930	4817	12230	12885
Total:	**0.284896**	**0.952393**	**25884**	**90854**	**132437**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

The kNN method with the temporal aspect has a threshold $h = 20$. An example of the computed kNN-distance on driver ID56 for this model configuration can be seen in Figure 4.14.

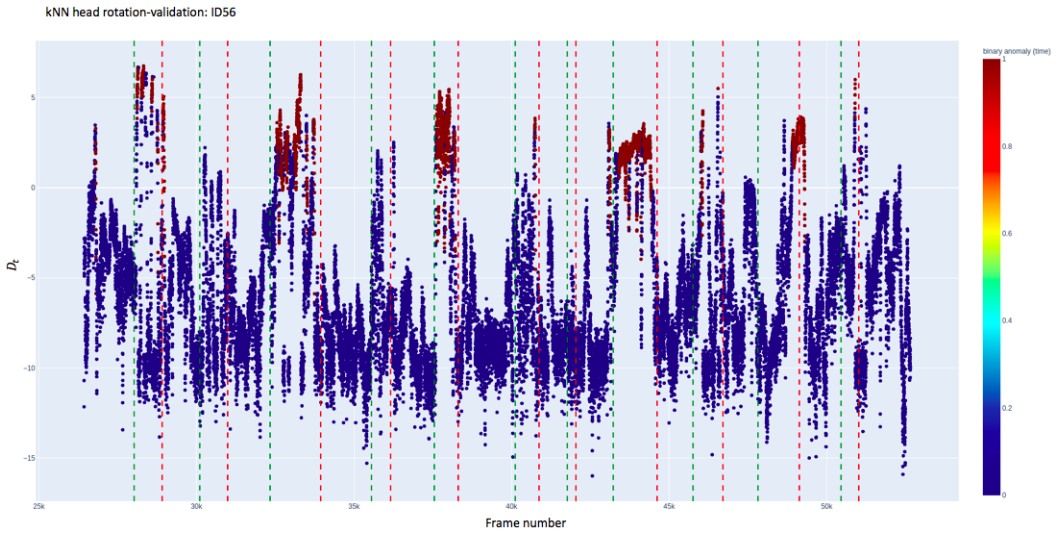

Figure 4.14: Plot of the kNN-distance on the validation set of ID56 with head rotation as feature, with temporal aspect included.

With the time aspect included, the model can obtain many anomalies in the anomalous scenarios. The total accuracy, total anomaly- and normal frames for this model configuration on the validation-set can be seen in Table 4.14.

Table 4.14: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head rotation, with the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.303557	0.999481	2586	8519	17317	17326
2	0.140917	0.981925	1380	9793	10648	10844
3	0.276215	0.985949	4586	16603	11227	11387
4	0.068655	0.986273	412	6001	12214	12384
5	0.344771	0.954068	3145	9122	14000	14674
6	0.031812	0.974784	274	8613	13298	13642
7	0.000000	0.997164	0	3078	10547	10577
8	0.442965	0.912159	4124	9310	11080	12147
9	0.115455	0.945060	756	6548	12626	13360
10	0.132189	0.982301	1117	8450	9657	9831
11	0.132448	0.961583	638	4817	12390	12885
Total:	**0.209324**	**0.9708536**	**19018**	**90854**	**135004**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.2.3 Head position, head rotation and gaze direction

Under this section the features head position, head rotation and gaze direction has been used for the kNN method with and without the temporal aspect. A visualization of the computed kNN-distance on driver ID2 for the this model without the temporal aspect can be seen in Figure 4.15.

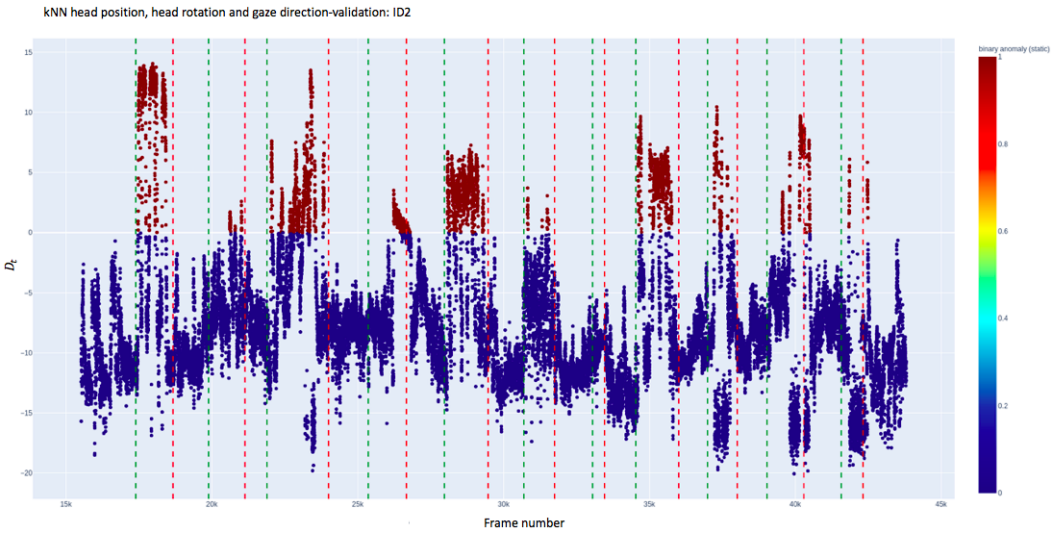

Figure 4.15: Plot of the kNN-distance on the validation set of ID2 with head position, head rotation and gaze direction as features, without temporal aspect included.

By looking at Figure 4.15, this model configuration detects the majority of the points in the anomalous scenarios and the normal scenarios does not detect a lot of anomalies. The total anomalies and normal frames under each scenario, including the normal- and anomaly accuracy can be seen in Table 4.15.

Table 4.15: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head position, head rotation and gaze direction, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.400634	0.980319	3413	8519	16985	17326
2	0.195139	0.969568	1911	9793	10514	10844
3	0.433837	0.968824	7203	16603	11032	11387
4	0.237960	0.976502	1428	6001	12093	12384
5	0.539027	0.942279	4917	9122	13827	14674
6	0.144085	0.972585	1241	8613	13268	13642
7	0.042235	0.969557	130	3078	10255	10577
8	0.501611	0.875772	4670	9310	10638	12147
9	0.215791	0.878443	1413	6548	11736	13360
10	0.195740	0.950870	1654	8450	9348	9831
11	0.233755	0.958246	1126	4817	12347	12885
Total:	**0.320360**	**0.949560**	**29106**	**90854**	**132043**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

The kNN method with the temporal aspect has a threshold $h = 20$. The computed kNN-distance for this model configuration on driver ID2 can be seen in Figure 4.16.

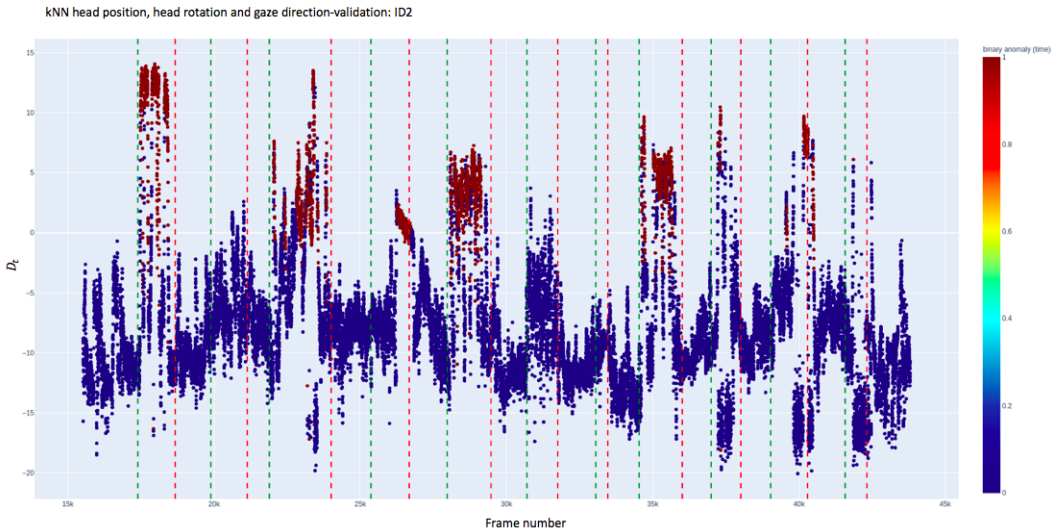

Figure 4.16: Plot of the kNN-distance on the validation set of ID2 with head position, head rotation and gaze direction as features, with temporal aspect included.

As can be seen in Figure 4.16, many of the points are obtained for the anomalous scenarios 1, 3, 5 and 8. These anomalous scenarios corresponds to look behind the shoulder, taking off the jacket, fight and vomit. The total results for this model configuration can be seen in Table 4.16.

Table 4.16: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head position, head rotation and gaze direction, with the temporal aspect included

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.357202	0.991516	3043	8519	17179	17326
2	0.122026	0.979251	1195	9793	10619	10844
3	0.374029	0.984719	6210	16603	11213	11387
4	0.200633	0.985546	1204	6001	12205	12384
5	0.481802	0.959725	4395	9122	14083	14674
6	0.071752	0.979548	618	8613	13363	13642
7	0.015595	0.993004	48	3078	10503	10577
8	0.461010	0.905820	4292	9310	11003	12147
9	0.141570	0.896856	927	6548	11982	13360
10	0.141893	0.972332	1199	8450	9559	9831
11	0.198464	0.961506	956	4817	12389	12885
Total:	**0.265117**	**0.964338**	**24087**	**90854**	**134098**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.2.4 Head position, head rotation and mouth points

Finally for the statistical method kNN, the feature combinations head position, head rotation and mouth points was used. The threshold for the kNN method both with and without the temporal aspect was set to $\epsilon = 6$. A visualization of the computed kNN-distance on driver ID1 can be seen in Figure 4.17.

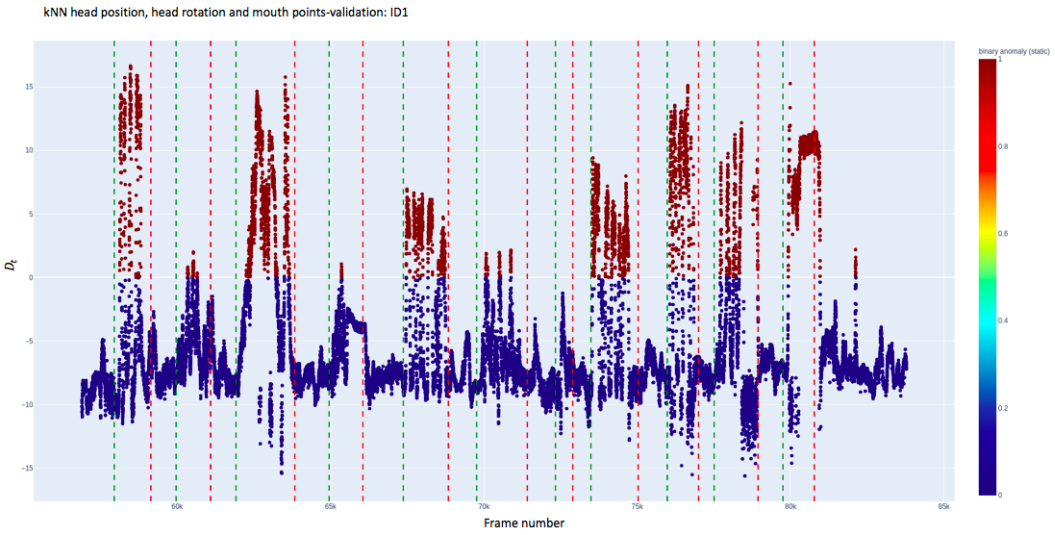

Figure 4.17: Plot of the kNN-distance on the validation set of ID2 with head position, head rotation and mouth points as features, without temporal aspect included.

For this model configuration, clear peaks of varying volume can be obtained by looking in Figure 4.17. The total frames obtained as anomalies and normal frames and the accuracy for detecting anomalies and normal data can be seen in Table 4.17.

Table 4.17: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head position, head rotation and mouth points, without the temporal aspect included.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.385022	0.993305	3280	8519	17210	17326
2	0.083835	0.981003	821	9793	10638	10844
3	0.444980	0.987354	7388	16603	11243	11387
4	0.104816	0.982962	629	6001	12173	12384
5	0.407257	0.960474	3715	9122	14094	14674
6	0.068037	0.979548	586	8613	13363	13642
7	0.016894	0.995935	52	3078	10534	10577
8	0.389044	0.908208	3622	9310	11032	12147
9	0.189065	0.936602	1238	6548	12513	13360
10	0.160592	0.975384	1357	8450	9589	9831
11	0.185593	0.964067	894	4817	12422	12885
Total:	**0.259559**	**0.969465**	**23582**	**90854**	**134811**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

For the kNN method with the temporal aspect, a threshold was set to $h = 20$. An example of the detected anomalies in the validation set of driver ID1 can be seen in Figure 4.18.

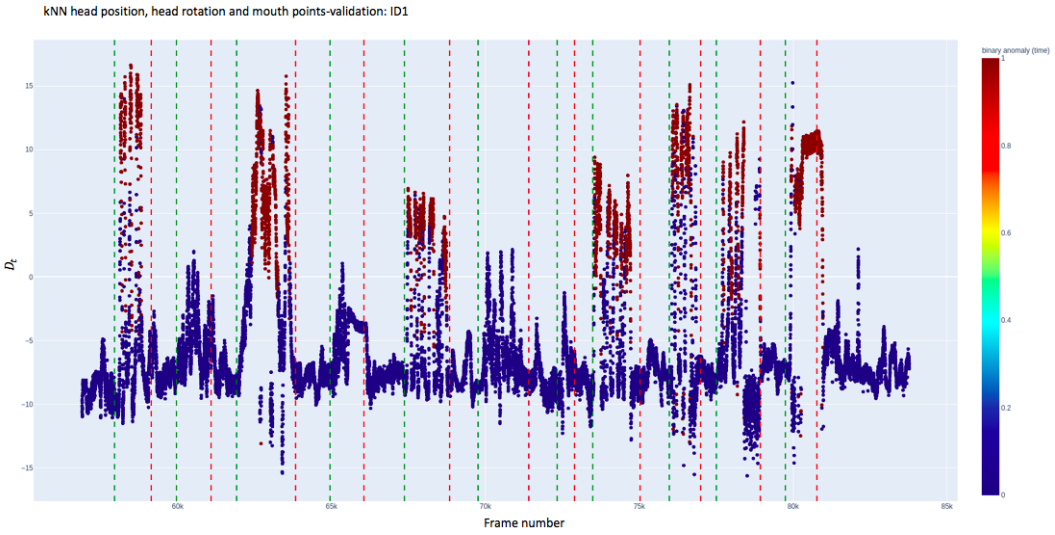

Figure 4.18: Plot of the kNN-distance on the validation set of ID1 with head position, head rotation and mouth points as features, with temporal aspect included.

By looking at Figure 4.18, the kNN model configuration with the temporal aspect had a hard time detecting the anomalous scenarios 2, 4, 6 and 7. But for the rest of the anomalous scenarios, the model could obtain many points as anomalies. The total accuracy and the corresponding total amount of anomaly- and normal frames for each scenario can be seen in Table 4.18.

Table 4.18: Accuracy and total number of frames identified as anomalies and normal behavior for the kNN-distance method using head position, head rotation and mouth points, with the temporal aspect included.

Scenario	**A. acc.**	**N. acc.**	**A. f.**	**Tot. f. A.**	**N. f.**	**Tot. f. N.**
1	0.371992	0.996306	3169	8519	17262	17326
2	0.060247	0.996496	590	9793	10806	10844
3	0.409625	0.989110	6801	16603	11263	11387
4	0.078654	0.994751	472	6001	12319	12384
5	0.332931	0.963814	3037	9122	14143	14674
6	0.034831	0.983653	300	8613	13419	13642
7	0.003249	0.997542	10	3078	10551	10577
8	0.327712	0.924096	3051	9310	11225	12147
9	0.122786	0.945808	804	6548	12636	13360
10	0.130296	0.986064	1101	8450	9694	9831
11	0.156944	0.964144	756	4817	12423	12885
Total:	**0.221135**	**0.976153**	**20091**	**90854**	**135741**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

4.3 Transformer

The transformer model is only trained with the features head position, head rotation and mouth position. The training loss, validation loss and an example of the detected anomalies on one driver id are visualized. Also the accuracy for the model is shown.

4.3.1 Head position, head rotation and mouth position

The transformer model was trained over 10 epochs with MSE being used as loss function. The training and validation loss is displayed in Figure 4.19.

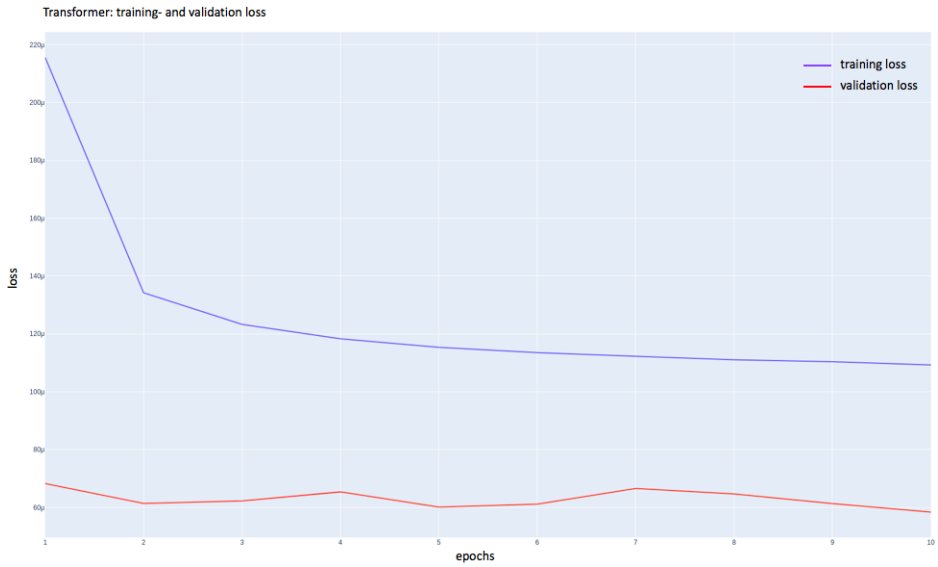

Figure 4.19: Validation- and training loss on the transformer with features as head position, head rotation and mouth points.

The model was then used to analyze the recorded validation videos, and an example of the model's performance on driver ID1 is displayed in Figure 4.20. The model is using a threshold for the detection of anomalies based on the reconstruction error $\epsilon = 0.03$, which is explained in more detail in 3.5.3 Performance test.

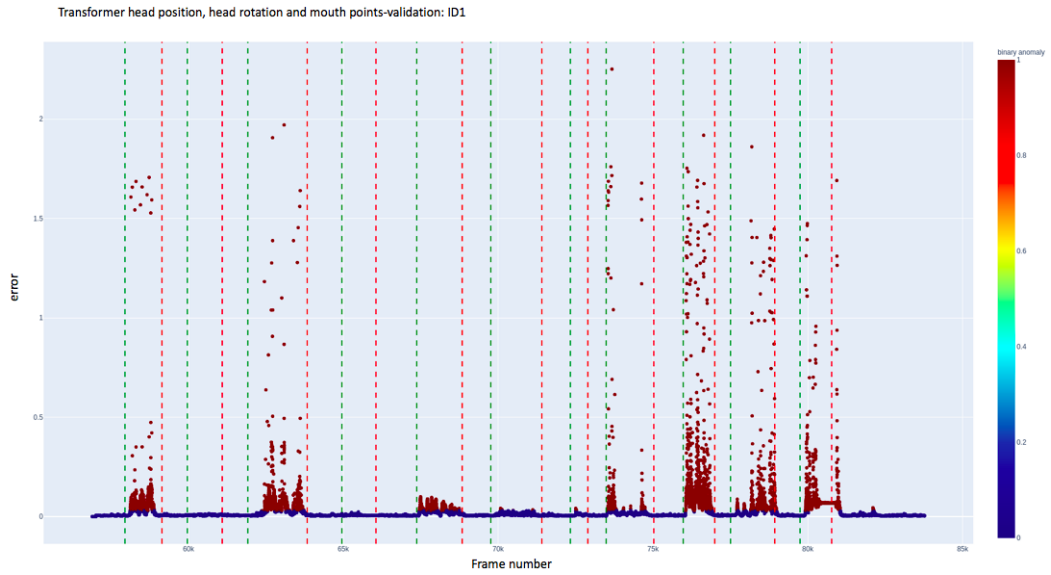

Figure 4.20: Plot of the reconstruction error on the validation set of ID1.

From Table 4.19 the transformer model performs great when it comes to identifying anomalies, with some scenarios being identified with up to almost 60% accuracy.

Table 4.19: Accuracy and total number of frames identified as anomalies and normal behavior for the transformer model.

Scenario	A. acc.	N. acc.	A. f.	Tot. f. A.	N. f.	Tot. f. N.
1	0.691396	0.981762	5890	8519	17010	17326
2	0.008271	0.985153	81	9793	10683	10844
3	0.596639	0.999737	9906	16603	11384	11387
4	0.049158	0.997901	295	6001	12358	12384
5	0.522254	0.962587	4764	9122	14125	14674
6	0.157785	0.977496	1359	8613	13335	13642
7	0.046459	0.996218	143	3078	10537	10577
8	0.072180	0.955133	672	9310	11602	12147
9	0.588271	0.953368	3852	6548	12737	13360
10	0.228047	0.992880	1927	8450	9761	9831
11	0.434918	0.936515	2095	4817	12067	12885
Total:	**0.341030**	**0.975132**	**30984**	**90854**	**135599**	**139057**

* A. acc. - Anomaly accuracy, N. acc. - Normal accuracy, A. f. - Anomaly frames amount, Tot. f. A. - Total frames anomaly, N. f. - Normal frames amount, Tot. f. N. - Total frames normal

5
Discussion

The following chapter includes an analysis of the results presented in the previous section, as well as comparisons between the different models based on these aforementioned results. Potential future work will also be discussed.

5.1 Analysis of results

The Mahalanobis method was performed with the following different feature combinations:

- Head position, head rotation & gaze direction,
- Head position, head rotation & mouth points,
- Head position,
- Head rotation,
- Gaze direction.

These were all tested both with and without temporal aspect being taken into consideration. From the tables describing the accuracy of these configurations, a continuous pattern seems to be that most model configurations appear to excel at anomaly detection using static classification, i.e. not taking the temporal aspect into account. This can however be caused by the temporal parameters not being optimized, as it is still theorized that the flexibility that comes with including a temporal aspect should result in more accurate anomaly predictions. Overall, all models seem to be performing about equally good when it comes to both the detection of anomalous scenarios as well as normal scenarios, except for the Gaze Direction. In Table 4.3 it can be seen that the model is more prone to find false positives in this case compared to the other feature combinations. The reason for this is because the tracking of this feature is often lost, resulting in bad quality measurements. This is clearly visible in Figure 4.3, where a "belt" of points can be seen at $D \approx 2.3$. This corresponds to the Mahalanobis distance calculated on the gaze, when it defaults to zero during periods of lost tracking. This obviously affects the model negatively, and is probably part of the reason why the model using head position, head rotation and gaze direction is just performing marginally better than the other models. The outcome of these models resulted in gaze direction being dropped as a separate feature for models in the other two methods.

The kNN-method was done with the following feature combinations:

- Head position, head rotation & gaze direction,
- Head position, head rotation & mouth points,

- Head position,
- Head rotation.

All these feature combinations were tested both with and without the temporal aspect. By looking at the normal- and anomaly accuracy between the models with and without the temporal aspect, the static classifications seems to detect the anomalous scenarios better. The difference in percentage points are approximately 5. This may occur since the parameters has not being optimized enough. The values of D_t are very noisy most of the time, which makes it hard to find the most optimal parameters. The normal accuracy percentage seems to be high on all the model configurations, but the models with the temporal aspects performs better to classify the points that are normal driver behavior. That occurs since there is a trade off depending on the value of the threshold. The higher value it is, the less points will be detected as anomalies in the anomalous scenarios. On the other hand, it will result in a higher percentage for the points that are detected as normal under the normal scenarios. The worst performance on the kNN methods was the feature combination head position, head rotation and mouth points. This may be because the mouth points are measured in pixel-position and has a other order of magnitude and will therefore affect the calculations a lot more than the other features named. The best model configuration for the kNN method is the model with features as head position, head rotation and gaze direction. The accuracy for both the detected anomalies and normal frames are high, as can be seen in Table 4.15. This model detects 32.04% of the anomalies in the anomalous scenarios and 94.96% of the normal in the normal scenarios.

Finally, the Transformer model was trained with the feature combinations:

- Head rotation, head position & mouth points

Due to the focus on the statistical methods, as well as the time needed to train the transformer models, only one feature combination was considered. The feature combination chosen was based on the implied superiority of this specific feature combination from the implemented statistical methods. From Table 4.19, it can be seen that the transformer performs really well when it comes to identifying anomalous scenarios, especially scenarios 1,3,5 and 9, which all have a frame accuracy over 50%. When compared to the other models, however, it is a bit more prone to labeling false positives during the normal scenarios. The reason for this is probably due to the threshold ϵ of the final model being optimized in a way which led to the model excelling at identifying the anomalous scenarios, but in the process also reporting more false positives. A possible solution could be to identify a new slightly higher threshold, and like the statistical models introduce an accumulating error variable for defining anomalies. By doing this, a more robust model would probably be achieved when it comes to not reporting too many false positives.

Overall the statistical methods and the transformer network performed good for all the different feature combinations, anomalous scenarios with a lot of movements, i.e. taking off the jacket and look behind the shoulder has a high percentage of detected anomalies. In most of the cases over 40% of the total points are detected as anomalies for these scenarios. On the other hand, for the anomaly scenario shocked,

all the models detected less amount of anomalies. A likely reason for this is due to the fact that these latter scenarios differ less from "normal driving" when considering the features being used.

When analyzing the results for a project like this, the quality and viability of the data need to be considered. To begin with, the provided data used as normal driving during the training consisted of 30 different IDs driving to and from Varberg and Gothenburg. This means that the majority of the available data consists of highway driving. The definition of normal driving behavior is obviously dependent on the environment a person is driving in. City driving and highway driving are for example very different when it comes to how the driver needs to act. When driving in a city, the driver will naturally need to check the surroundings more often, look in the mirrors as well as keeping track of more cars and pedestrians compared to when they are driving on a highway, resulting in different distributions of the data in these scenarios. For our trained models, this means that they have all been trained on recognizing normal driver behavior during highway driving. Due to this fact, if the models were to be applied to a scenario of city driving, the models would probably identify more anomalies than preferred. When it comes to the collected validation data, these videos were all recorded in a garage, where the test subject needed to imagine themselves driving in the different defined scenarios. This obviously means the authenticity of the data needs to be questioned, and is treated as a limitation in this project. The results derived from applying the models to these validation sets need to be analyzed with this in mind as well. For example, during the scenarios of normal driving, some test subjects forgot that they needed to imagine they were still driving, which led to some passages of normal driving where the person took their hands of the wheel and looked back in the backseat to get instructions about the next anomalous scenario coming up. This probably affected the performance of the models on the normal scenarios.

Another aspect that needs to be considered is the fact that different drivers have different behavioral patterns, which all could be labeled "normal" from an outside perspective. This obviously makes it hard for any kind of model to correctly classify a random drivers behavior. The approach in this project has resulted in a number of models where the drivers initial or "normal" starting position has an effect on the model performance. This is especially true in the statistical models, since they both classify the frames based on values of absolute position of the features in that specific frame. This means that in theory, a very tall person could probably be classified as behaving abnormally in their initial "normal" position, due to the head position being significantly different compared to most other people. This could be one reason the transformer manages to slightly outperform the statistical models, since this method takes into consideration the changes in the features over a span of time, resulting in relative movement from earlier positions being considered by the model.

Some model configurations obtain a frame as an anomaly or outlier if one of the data-points has bad quality. This means that if the model is based on mouth-points

and a driver covers the mouth with the hand, the model signals the image as an anomaly. Broadly speaking, this means that a normal driver scenario can detect some of the frames as anomalies.

5.2 Future work

Time can be spent to optimize the parameters more in the statistical methods, especially the parameter h for the models with temporal aspect. The parameters for the transformer model can also be more optimized to obtain a clearer difference between the normal- and anomaly driver behavior. Another thing to work on could be trying out new features and feature combinations for the models, to see if that could yield better results in some cases.

Another suggestion to work further with this project is to set a higher threshold h for the models with temporal aspect. Instead of calculating the percentage of anomalies obtained in total for each scenario, a threshold is set to obtain only one or a few frames as anomalies under the anomalous scenarios. If the model can detect one anomaly frame for each scenario, it is enough to classify that scenario as an anomaly, if the threshold h is high enough. If this got tuned well enough, the main benefit would be that the models would overall be more robust when it comes to not reporting false positives. With this higher threshold, the results achieved from the models when classifying frames during normal scenarios hints at this being the case.

The way the models are operating, every single frame gets classified as either being normal or anomalous. To incorporate this kind of system in real time in a car, that would probably not be a feasible way to report anomalous behavior. One way to solve this could be to introduce a sliding window in the different methods, which always looks at a fixed interval of the N most recent frames. The system could then try to identify and report anomalous behavior based on if some defined portion of these last N frames are classified as anomalies. This could also potentially act as a valid method for incorporating a temporal aspect into the static models.

One more interesting thing to go forward with could be to adjust the way the feature metrics work. As was mentioned in the previous part, all features are measured in a world coordinate system, resulting in all features being measured in absolute position. This makes it harder for the model to classify different persons correctly, due to them being significantly taller than the average person for example. A way to solve this could be to normalize the features based on an initial "normal" position of a driver, allowing the features to then express the relative change from this initial position. This would mean that the values of the features would be more similar between different drivers, even if they are very different when it comes to stature. This could be taken even further, by training unique models for every person, based on their specific behavior. This would at least be interesting to do, to see if the results would increase significantly from this.

6
Conclusion

To conclude this project, the questions posed in the initial part of the project is answered in this section.

Are deep learning or statistical methods suitable for detecting unusual driver behavior considering performance and execution time?

From the different model configurations used during this project, the model based on deep machine learning, or more exactly a transformer network performed best for detecting unusual driver behavior considering both performance and execution time. The main reason that the statistical models are not performing as well as the transformer is probably due to the data not being very varied. By adding more data of different types of driving, as well as including parameters to make the methods more robust towards different persons would improve the performance.

Can driver anomaly detection be based on spatial information alone or does temporal information need to be exploited for robustness?

Based on the fact that the transformer model performed the best, this implies that temporal information being included increases the robustness of the models. As was mentioned, it is still believed that temporal information with optimized parameters would increase the performance for the statistical methods as well. Also, as some anomalous scenarios are directly dependent on the time something is being performed (e.g. looking over the shoulder), it makes sense that temporal information needs to be included.

Are output data provided by driver monitoring and cabin monitoring systems sufficient for detecting anomalies in driver behavior?

This project has mainly concerned observing features connected to the head of the driver. For the defined anomalous scenarios in this project, relatively satisfactory results are obtained, which points at this kind of information being sufficient. By also putting more effort into utilizing cabin monitoring and observation of body pose of the driver, this could potentially increase the performance further.

Can driver anomaly detection be accomplished by unsupervised learning or are labeled events necessary?

In this project, unsupervised learning has been used for all implemented methods, with relatively good results, which points at unsupervised learning potentially being sufficient. It could even be argued that unsupervised learning is necessary for this kind of project, since labeling anomalous scenarios correctly and with enough

generality is very hard.

What can be defined as anomalies in driver behavior?

This was one of the harder parts in this project. An anomalous behavior of a driver is subjective, and in this project it was defined as when the driver looses focus of the road, due to distractions or decreased mental or physical health. With this definition a couple of fairly general and unique anomalous scenarios were created to test the implemented methods. Even though this definition is far from entirely encompassing of what an anomalous behavior is, the generality of the definition served the purpose of evaluating the methods, and implies that similar results could be achieved for other definitions of what an anomalous behavior is.

Bibliography

[1] Keith D. Foote, *A Brief History of Big Data*, 2019. [Online] Available: https://www.dataversity.net/brief-history-big-data/#, picked: 10-02-2021

[2] Gali Halevi, MLS, PhD & Dr. Henk F. Moed, *The Evolution of Big Data as a Research and Scientific Topic: Overview of the Literature*, 2012. [Online] Available: https://www.researchgate.net/publication/285119834_The_evolution_of_big_data_as_a_research_and_scientific_topic_Overview_of_the_literature, picked: 10-02-2021

[3] Jun Wu, *AI, Machine Learning, Deep Learning Explained Simply*, 2019. [Online] Available: https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960, picked: 10-02-2021

[4] Susan Li, *Anomaly Detection for Dummies*, 2019. [Online] Available: https://towardsdatascience.com/anomaly-detection-for-dummies-15f148e559c1, picked: 10-02-2021

[5] Smart Eye AB, *About us*, 2021. [Online] Available: https://smarteye.se/about-us/, picked: 10-02-2021

[6] V. Bajaj. *Unsupervised Learning For Anomaly Detection*, 2020. [Online] Available: https://towardsdatascience.com/unsupervised-learning-for-anomaly-detection-44c55a96b8c1, picked: 30-03-2021

[7] S. Mehta, P. Kothuri, D. Garcia. *Anomaly Detection for Network Connection Logs*, 2020. [Online] Available: https://arxiv.org/pdf/1812.01941.pdf, picked: 30-03-2021

[8] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, J. Sun†. *Anomaly Detection for a Water Treatment System Using Unsupervised Machine Learning*, 2017. [Online] Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8215783, picked: 30-03-2021

[9] Q. Jan, J. Chen, L. De Strycker. *An Outlier Detection Method Based on Mahalanobis Distance for Source Localization*, 2018. [Online] Available: https://www.researchgate.net/publication/326282908_An_Outlier_Detection_Method_Based_on_Mahalanobis_Distance_for_Source_Localization, picked: 30-03-2021

[10] X. Gu, L. Akoglu, A. Rinaldo. *Statistical Analysis of Nearest Neighbor Methods for Anomaly Detection*, 2020. [Online] Available: https://papers.nips.cc/paper/2019/file/805163a0f0f128e473726ccda5f91bac-Paper.pdf, picked: 30-03-2021

[11] Romero Morais, Vuong Le, Truyen Tran, Budhaditya Saha, Moussa Mansour, Svetha Venkatesh *Learning Regularity in Skeleton Trajectories for Anomaly Detection in Videos*, Applied Artificial Intelligence Institute, 2019. [Online] Available: https://arxiv.org/pdf/1903.03295.pdf, picked: 20-02-2021

[12] P. C. Mahalanobis: On the generalized distance in statistics. Proceedings of the National Institute of Sciences (Calcutta), 1936, 2, pp. 49–55. Picked: 25-02-2021

[13] FACTA UNIVERSITATIS (NIS) ˇ Ser. Math. Inform. Vol. 34, No 3 (2019), 583–595 https://doi.org/10.22190/FUMI1903583G, picked: 25-02-2021

[14] Geek3, *Chi-square distribution plots for varying degrees of freedom*, 2010. [Online] Available: https://commons.wikimedia.org/wiki/File:Chi-square_pdf.svg, picked: 20-02-2021

[15] R. Kissell, J. Poserina, Chapter 4 - Advanced Math and Statistics, Editor(s): Robert Kissell, Jim Poserina, Optimal Sports Math, Statistics, and Fantasy, Academic Press, 2017, Pages 103-135, ISBN 9780128051634, https://doi.org/10.1016/B978-0-12-805163-4.00004-9. Picked: 15-03-2021

[16] O. Harrison. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*, 2018. [Online] Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-\6a6e71d01761, picked: 30-03-2021

[17] M. Mozaffari, Y. Yilmaz. *Online Multivariate Anomaly Detection and Localization for High-dimensional Settings*, 2020. [Online] Available: https://arxiv.org/pdf/1905.07107.pdf, picked: 30-03-2021

[18] J. Brownlee, *What is Deep Learning?*, 2019. [Online] Available: https://machinelearningmastery.com/what-is-deep-learning/, picked: 03-03-2021

[19] J. Schmidhuber, *Deep Learning in Neural Networks: An Overview*, 2014. [Online] Available: https://arxiv.org/pdf/1404.7828.pdf, picked: 02-02-2021

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin. *Attention Is All You Need* 2017. [Online] Available: https://arxiv.org/pdf/1706.03762.pdf, picked: 07-03-2021

[21] *How Transformers Work*, 2019. [Online] Available: https://towardsdatascience.com/transformers-141e32e69591, picked: 05-03-2021

[22] Hopkins, Mark and Meyer, Joe. *Generating Free Verse Poetry With Transformer Networks*, 2019. doi: 10.13140/RG.2.2.34638.89924, picked: 2021-04-25

[23] H. Song, D. Rajan, J. J. Thiagarajan, A. Spanias. *Attend and Diagnose: Clinical Time Series Analysis using Attention Models*, 2017. [Online] Available: https://arxiv.org/pdf/1711.03905.pdf, picked: 2021-04-07

[24] R.De Maesschalck, D. Jouan-Rimbaud, D.L. Massart, *The Mahalanobis distance*, 2000. [Online] Available: https://www.sciencedirect.com/science/article/abs/pii/S0169743999000477, picked: 16-02-2021

[25] Dapeng Li, in Comprehensive Geographic Information Systems, *GIS Methods and Techniques*, 2018. [Online] Available: https://www.sciencedirect.com/topics/computer-science/euclidean-distance, picked: 19-02-2021

[26] CHANDOLA, V., BANERJEE, A. & KUMAR, V. (2009) *Anomaly detection: A survey", ACM Computing Surveys* (CSUR), vol. 41, no. 3, pp. 1-58. Picked: 20-02-2021

[27] M. Alam. *Statistical techniques for anomaly detection*, 2020. [Online] Avaliable: `https://towardsdatascience.com/statistical-techniques-for-anomaly-detection-6ac89e32d17a`, picked: 07-03-2021

A

Appendix 1

Recordings to Master Thesis project
Anomaly detection of driver behaviour

Klas and Pontus

March 2021

Guide for the recording

Different scenarios will be recorded, where each of these are described in this document. The recording will start with the test subject pretending to "drive normally" for 30 seconds. This will include sitting in a normal driving position, with both hands on the wheel and eyes on the road, while also occasionally looking in the rear mirrors. The anomalous scenarios will then be performed in succession, with 10-20 seconds of normal driving in between.

- The videos will only be used by the Master Thesis students and Smart Eye AB.

Scenarios

The following scenarios will be performed and a description of how they are done is stated under every scenario.

Normal

The person will drive the car with a normal behaviour. In other words, with the eyes pointed on the road.

Look behind the shoulder

When the recording starts the person will look behind the shoulder to the right (back in the back seat) for about three seconds.

Drunk

When the recording starts the person will swing back and forth with the upper body and head, also the eyes should look tired with "flickering gaze" and "droopy eyelids".

Taking off the jacket

In this scenario, the person will take off the jacket. Start in a normal position and take off the seat-belt, remove your jacket and put it in the back seat. The person will imagine still driving the car, so trying to have eyes on the road and at least one hand on the wheel during this scenario.

Heart-attack

When the recording starts the driver enters a state of respiratory distress where he/she is gasping for breath for five seconds, and after that falling back in the seat with open eyes and open mouth.

Fight

When the recording starts the driver will be looking to the right, and be in a "heated argument" with the passenger for five seconds.

Panic-attack

When the recording starts the driver will feel stressed. Move quickly with the head in different directions while shaking.

Shocked/Scared

The driver starts off "normally" with eyes on the road, then he/she opens the mouth and eyes widely to simulate the feeling of being shocked.

Change playlist on phone

In this scenario you will pick up your phone and place it on your knee. After that, you will imagine changing/starting a song on the phone, while still driving the car.

Vomit

When the recording starts the driver moves his/her head to the side and starts retching for five seconds.

Sleep

At the start, be in the normal drive-position for five seconds with "droopy eyelids", then falling asleep by letting the head drop forward and closing their eyes.

Dead

Start in a normal position and then move your head forward with closed eyes.

Thanks

Thank you for being a part of our Master Thesis project!

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY