DEGREE PROJECT IN VEHICLE ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Automated advanced analytics on vehicle data using AI

SIMIN ZHANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Automated advanced analytics on vehicle data using AI

Simin Zhang

Master of Science in Engineering
Degree project in Vehicle Engineering
KTH Royal Institute of Technology

Supervisor at (Scania): Paola Maggino and Nikhil Thakrar
Supervisor at KTH: Mikael Nybacka
Examiner at KTH: Mikael Nybacka

Date of presentation: 2020-10-21

TRITA SCI GRU 2020:337

KTH Royal Institute of Technology
School of Engineering Sciences
KTH SCI SE-100 44 Stockholm, Sweden
URL: http://www.kth.se/sci

Abstract

The evolution of electrification and autonomous driving on automotive leads to the increasing complexity of the in-vehicle electrical network, which poses a new challenge for testers to do troubleshooting work in massive log files. This thesis project aims to develop a predictive technique for anomaly detection focusing on user function level failures using machine learning technologies.

Specifically, it investigates the performance of point anomaly detection models and temporal dependent anomaly detection models on the analysis of Controller Area Network (CAN) data obtained from software-in-loop simulation. For point anomaly detection, the models of Isolation forest, Multivariate normal distribution, and Local outlier factor are implemented respectively. For temporal dependent anomaly detection, the model of an encoder-decoder architecture neural network using Long Short-Temporal Memory (LSTM) units is implemented, so is a stacking hybrid detector in the combination of LSTM Encoder and Local outlier factor.

With a comparison of the comprehensive performance of the proposed models, the model of LSTM AutoEncoder is selected for detecting the anomalies on sequential data in CAN logs. The experiment results show promising detection performance of LSTM AutoEncoder on the studied functional failures and suggest that it is possible to be deployed in real-time automated anomaly detection on vehicle systems.

Keywords

Anomaly detection, Machine learning, Controller Area Network, User function level failure

Sammanfattning

Utvecklingen av elektrifiering och autonom körning på fordon leder till den ökande komplexiteten i fordonets elektriska nätverk, vilket utgör en ny utmaning för testare att göra felsökningsarbete i massiva loggfiler. Detta avhandlings syftar till att utveckla en förutsägbar teknik för detektering av avvikelser med fokus på användarfunktionsnivåfel med maskininlärningstekniker.
Specifikt undersöker den prestandan hos punktavvikelsedetekteringsmodeller och tidsberoende anomalidetekteringsmodeller på analysen av data från Controller Area Network (CAN) erhållen från simulering av mjukvara in-loop. För detektion av punktavvikelser implementeras modellerna för Isolation forest, Multivariate normal distribution och Local outlier factor. För temporär beroende anomalidetektering implementeras modellen för ett kodnings-avkodningsarkitekturneuralt nätverk som använder Long Short-Temporal Memory (LSTM) -enheter, så är en stapling hybriddetektor i kombination med LSTM Encoder och Local outlier factor.
Med en jämförelse av den omfattande prestandan hos de föreslagna modellerna väljs modellen för LSTM AutoEncoder för att detektera avvikelser på sekventiell data i CAN-loggar. Experimentresultaten visar lovande detektionsprestanda för LSTM AutoEncoder på de studerade funktionella misslyckandena och föreslår att det är möjligt att distribueras i realtid automatiserad anomalidetektering på fordonssystem.

Nyckelord

Avvikelsedetektion, Maskininlärning, Controller Area Network, Fel på användarfunktionsnivå

Acknowledgements

This Master Thesis was carried out at the department of EPIA Embedded process tools development A at Scania CV AB in Södertälje, Sweden and at Department of Vehicle Engineering and Solid Mechanics at Royal Institue of Technology (KTH) in Stockholm, Sweden.

I would like to thank my supervisors at Scania, Paola Maggino and Nikhil Thakrar for their help and constructive ideas throughout the entire project, as well as Simon Milvert for his guidance on user function and software-in-loop simulation. I would also thank my manager at Scania, Gunnar Robertsson for generous encouragement and support on my experience at Scania, so as my colleagues in EPIA department.

Beside, I would also thank associate professor Mikael Nybacka and Lars Drugge at KTH for their incredible support in my thesis work.

Finally, I would specially thank my family for their care and support during my entire master program, especially in the Covid-19 period.

Acronyms

CAN	Controller Area Network
AI	Artificial Intelligence
ECU	Electronic Control Unit
LSTM	Long Short-Term Memory
RNN	Recurrent neural network

Contents

Chapter 1

Introduction

Nowadays, the top two most important evolution in the area of automotive electronics is electrification and autonomous driving, which have the world's expectation on sustainability, safety, and intelligence. However, as the electronic systems and in-vehicle communication become more complex, it also comes with a new requirement on the stability and reliability of them. The purpose of this thesis project is to develop a technique using artificial intelligence to detect real-time anomalies efficiently and accurately, focusing on functional failure, for better supporting vehicle systems.

1.1 Background

In the trend of the electrification era, Scania is committed to product development and evolution, with the increasing complexity of its electronic systems, which brings a new challenge to system reliability and mega data analytics. Also, there is a new need for developing a more advanced supporting tool to overcome these difficulties to accommodate the technological advancements in the field. Besides, the rise of autonomous driving also needs much stronger computation ability for vehicles since they will face more various and more complex working scenarios.

On the other hand, the development of Artificial Intelligence (AI) redefines industrial autonomy and brings a bright picture for their applications in the industrial field. As a data-driven technique, artificial intelligence can get the system the self-growth characteristics, which means it can update and optimize with data feeding. It has the advantage in big data analytics, through the cognition, learning, and inference to data, artificial intelligence can realize pattern recognition and prediction in big data with

high accuracy and efficiency, especially after the appearance of parallel processing which improves the computation ability of algorithms.

1.2 Motivation

The increasing complexity of vehicle electronics such as vehicle controllers and other smart components brings a new challenge to system computation and reliability, to the in-vehicle communication network due to the increase of workload to the data flow. These components are required to be monitored in real-time to support and realize complex functions in vehicles. And as a consequence, the massive log files, huge in numbers and quantity, will generate a heavy workload to testers with manual and time-consuming work on results analysis. Furthermore, this will also lead to the difficulty in troubleshooting since the diagnosis could need much more complex analysis of extensive data to identify problems.
Hence, there is an urgent need to find a solution to simplify the testing procedure and improve the testing capacity. Specifically, it should be able to do real-time analysis with data acquired from Electronic Control Unit (ECU), detect faulty behaviors, and record corresponding information, which can further help testers locate root causes more efficiently.

1.3 Methodology

Taking the complexity of vehicle electronics into consideration, it is hard for testers to do troubleshooting work. A great breakthrough is to monitor the situation of user functions. In the in-vehicle communication network, various user functions are sharing data from different sensors and hardware, so by detecting user function level failures it is theoretically possible to track back from the user function network to find specific problems. Besides, this integrated strategy can simplify testing procedures greatly.
Therefore, this thesis project is to develop a predictive technique to detect anomalies focusing on user function level failures, with the application of AI on vehicle data.

1.4 Research questions

In this thesis project, predictive models will be built with real-time data logged from Controller Area Network (CAN) for anomaly detection. Given the analysis of problems and methodology above, it is clear that the model needs to meet some certain criteria to function, which leads to some research questions to be addressed, i.e.

- What kind of model can detect anomalous behaviours?
- Which model have the best potential to be applied to different user functions?

1.5 Outline

In the next chapter, the theoretical background will be discussed, including some necessary theoretical knowledge for further chapters, as well as some related work on this topic. Furthermore, the method will include more technical details, such as the data preprocessing, the principles of models, and the strategy of performance evaluation. The results will then be discussed, with a conclusion regarding research questions and potential for future work following.

Chapter 2

Theoretical background

In this chapter, the necessary theoretical foundation for this project will be introduced, including the type of anomalies and appropriate machine learning approaches, so will related work on this topic to be discussed.

2.1 Type of anomaly

Anomalies are data points which show abnormal pattern when compared to those well-defined normal behaviors. In general, anomalies can be classified into the following three categories:

- Point anomaly
  If an individual data instance is far away from the rest, then it can be regarded as a point anomaly. It shows temporal independence with anomalous behavior in global [9].
- Contextual anomaly
  Contextual anomaly usually appears in temporal data, as a specific data instance that has a large difference between its behavior and those of its neighbors[4].
- Collective anomaly
  This type of abnormality is composed of a combination of multiple objects. There may be no abnormal behavior when looking at an individual alone, but when these individuals appear at the same time it constitutes an abnormality[4]. Collective anomalies appear in multiple ways of composition, may be composed of several single points, or maybe composed of several sequences.

It is worth mentioning that anomaly detection needs to adjust strategy and choose suitable algorithms according to the characteristics of the anomaly.

2.2 Type of machine learning

Machine learning has a significant influence on industrial development, especially in the area of data analysis thanks to its strong data-driven learning capacity. According to the type of problems to be solved, machine learning can be classified into three categories: supervised learning, unsupervised learning, and semi-supervised learning.

- Supervised learning
  Supervised learning is a way of using labeled training data to obtain an optimal model where labeled training samples means clear pairs of inputs and their corresponding outputs[19]. A good supervised learning model can map all the inputs to the corresponding outputs as correctly as possible, and have a strong ability to be generalized for unseen instances.

- Unsupervised learning
  Compared to supervised learning, unsupervised learning can discover hidden patterns or group data from unlabelled training samples, essentially it is a statistical method[3]. The two main types of unsupervised learning are cluster analysis and dimension reduction.

- Semi-supervised learning
  For many practical problems, labeled samples are hard or very expensive to collect while unlabeled samples are much easier to obtain. As an approach between supervised learning and unsupervised learning, Semi-supervised learning can make full use of a large number of unlabelled samples to improve learning performance with the pre-training using labeled samples, in this way it can avoid the waste of data resources and get rid of the dilemma when supervised learning lacks of generalization ability with insufficient labeled samples, and unsupervised learning is inaccurate without the guidance of labeled samples[3][33].

2.3 Related work

Anomaly detection has been popular research interest in the domain of artificial intelligence recently due to its wide application potential. For point anomalies detection, novelty or outlier, there are many classical machine learning algorithms widely applied, such as some density-based approaches (K-nearest neighbor[32], local outlier factor[2], and isolation forest[15]), and cluster analysis-based approaches (K-means[6]). For sequential anomalies detection in time-series signals, there are also many developed techniques, such as Hidden Markov models[5] and recurrent neural network[29].

As a continuation of the Scania thesis project 'Anomaly detection', Egill Vignisson who worked on the last topic 'Anomaly detection in a complex system using AI' extracted the top 9 the most related signals $S_2 \sim S_{10}$ from 1415 signals to one manually selected signal S_1 and built a recurrent neural network many-to-one model by mapping the 10 signals mentioned above $S_1 \sim S_{10}$ to signal S_1 to realize anomaly detection[29]. However, this manual feature selection would limit the model in such a specific failure which led to the lack of generalization ability.

To make up for this kind of shortcoming, AutoEncoder is a great unsupervised deep learning method in this field, by an encoder-decoder architecture to reconstruct data and monitor the reconstruction error to detect anomalies[3]. Therefore, Long Short-Term Memory (LSTM)-based AutoEncoder is an appropriate choice when taking the temporal dependency of time-series vehicle data into consideration, as the LSTM-based recurrent neural network has the advantage in processing sequential data and predicting which has been applied widely in spacecraft[10], automobile[27], and vessels[16].

Besides, dimension reduction is a feature engineering technique that should be taken into consideration because data from high-dimensional space may lead to undesired performance, even the curse of dimensionality, as raw data may introduce too much noise or irrelevant information[28]. Dimension reduction can make data analysis more accurate and more efficient in reduced space. The methods can be divided into linear (such as Principal Component Analysis[28]) and nonlinear approaches (such as AutoEncoder). In this project, the effect of dimension reduction on sequential data by AutoEncoder will be studied.

Chapter 3

Methods

In this chapter, methods used in this thesis project will be explained step by step in detail, including data acquisition, data preprocessing, modeling, and performance evaluation.

3.1 Data acquisition

As mentioned in section 1.3, this project is to develop an anomaly detection technique for user functions, for a well-defined user function, the inputs, and the outputs usually have a clear mapping relationship. But functional failure can lead to unexpected outputs even if given the same inputs, in other words, that is abnormal behaviour.
In this project, the user function engine speed control using accelerator pedal is selected as a case to do further experiments because both normal behavior and abnormal behavior of it are easy to understand. Data resources including both normal behavior and abnormal behavior are necessary to be collected for training models and evaluating the performance of them.
Data used in this project was from the software-in-loop simulation at a Scania simulation platform, raw CAN log files of normal driving, and abnormal driving with fault injection where some parameters for the user function were changed to different values for every logging to make a difference from the preset behavior. There are a total of 160 000 CAN messages reported in normal driving log files and 50 239 CAN messages reported in abnormal driving log files. However, the raw data consists of all message identifications which are defined in the CAN bus, so it is necessary to filter out useless information from raw data.

3.2 Data preprocessing

The quality of data directly determines the performance and the generalization ability of models, as too much useless information or inappropriate format for the data may lead to inefficient and bad performance. Therefore, data preprocessing is a very important part of data analysis. In this project, data preprocessing includes three steps: resampling and data cleaning, data normalization and data splitting.

3.2.1 Resampling and data cleaning

For the user function engine speed control using accelerator pedal, it has 13 CAN signals from 9 different identifications in the CAN bus, so the first step of work is to extract CAN signals from raw CAN log files according to their CAN identifications, start bits, and lengths so that the irrelevant messages can be filtered out and the valuable information can be extracted.
However, because CAN messages are reported in different frequencies based on their importance to the system, which leads to inconsistency of timestamps of CAN messages, there are null values for some signals in every timestamp. So it is necessary to interpolate to get a uniform format of 2 dimensions (samples, features). It is worth noting that the timestamps for each CAN log file are discontinuous, so the interpolation should be done file by file. Here the missing values for each signal are interpolated by the last observed value before that timestamp since it can be regarded as have not been updated in time series data. And for those in first rows with missing values at the beginning of logging, they have to be cleaned.

3.2.2 Data normalization

Data normalization is to scale data of every feature to a small specific interval separately before data analysis, removing the unit limit of the data and converting them into dimensionless pure values, so that data of different features with different units or magnitudes can be compared and weighed[11]. On the one hand, normalization can speed up the convergence of gradient descending by making all features on a similar scale[24]. On the other hand, it can also make a balance of the contribution of all features to results because the model may highlight the roles of features with high

values and lower the confidence of results if without normalization.
Two of the main normalization methods are min-max normalization and Z-score normalization which is also named standard normalization.

- min-max normalization
  This normalization can scale data of every feature in the range of $[0, 1]$ by applying equation 3.1.
  $$x_{normalization} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$
- Z-score normalization
  This normalization can make the processed data conform to the standard normal distribution by applying equation 3.2.
  $$x_{normalization} = \frac{x - \mu}{\sigma} \tag{3.2}$$
  where μ is the mean of all sample data, and σ is the standard deviation of all sample data.

3.2.3 Data splitting

To test the generalization ability of models and avoid overfitting, it is important to split the dataset into a training set, testing set, and validation set if necessary. To be more specific, using the data of training set to generate scale for normalization and train model, followed by using the data of validation set to tune parameters of the model, and then use the error on the test set as the generalization error of the final model in dealing with realistic scenarios[8][30].
As is known, labeled anomalous data are hard to collect due to the expensive and time-consuming process while labeled normal data are much easier to collect. Similarly, in this project, taking the imbalance of the number of positive samples and negative samples (positive samples are the data for abnormal driving and negative samples are those for normal driving) into account, unsupervised learning and semi-supervised learning are determined as the main methods to learn the hidden pattern in normal data and to detect anomalies in abnormal data.
Therefore, the data of normal driving is divided into two parts, 70% for the training set, and the rest to be mixed with the data of abnormal driving in a similar proportion for validation and testing. Something worth to mention is that the data splitting is

randomly sampling behind reshape because time series prediction needs to reshape the data into a temporal format (samples, timestamps, features). The data splitting strategy is shown in figure 3.2.1.

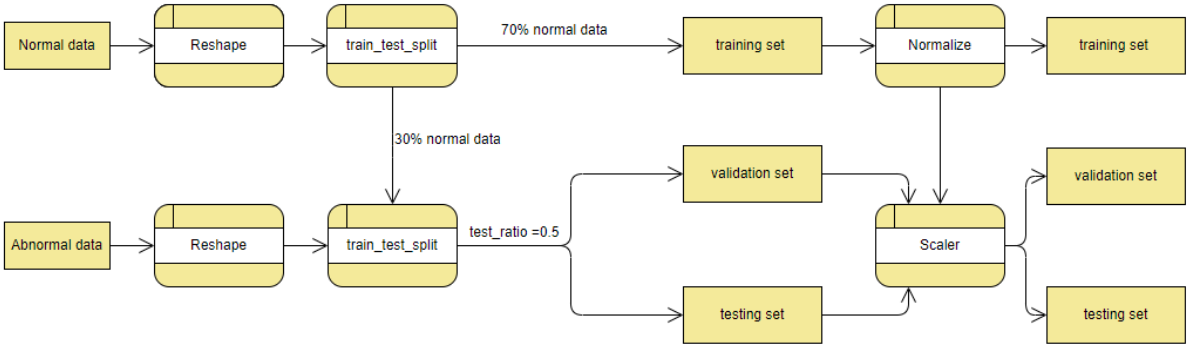

Figure 3.2.1: Data splitting strategy

3.3 Models

In this section the principles of different models used for anomaly detection will be discussed in detail, as shown in table 3.3.1. For point anomaly detection, isolation forest, multivariate normal distribution, and local outlier factor are built. Besides, the performance of LSTM AutoEncoder is also investigated in both data reconstruction and dimension reduction, focusing on the detection of the contextual anomaly and collective anomaly which are temporal dependent. The hybrid detector is an outlier detection model mentioned above with dimension reduced data processed by LSTM Encoder.

Table 3.3.1: Models for anomaly detection

Type of anomalies	Models
Point anomaly	Isolation forest
	Multivariate normal distribution
	Local outlier factor
Contextual anomaly/Collective anomaly	LSTM AutoEncoder
	Hybrid detector

3.3.1 Isolation forest

Isolation forest is a fast outlier detection method based on ensemble learning, with linear time complexity and high accuracy[15].

Just as random forest is composed of a large number of decision trees, isolation forest is also composed of a large number of isolation trees. However, the difference of isolation tree from decision tree is that the algorithm randomly selects attribute and splitting value for every node in building tree instead of based on the information entropy[15]. Assuming that there are N pieces of data $\{x_1, x_2, ..., x_N\}$ in a dataset, ψ pieces of data are uniformly and randomly sampled without replacement from the dataset as the training samples for building an isolation tree. A feature is randomly selected, and value is randomly selected within the range of all values of this feature (between the minimum and maximum values), and then the node has two branches, and the samples smaller than this value are divided into the nodes on the left, those greater than or equal to this value are divided to the right of the node[15]. In this way, two branches of a binary tree are built, and then the above process is repeated on the left and right data sets respectively until reaching the termination condition. There are two termination conditions, one is that the data itself cannot be subdivided (only one sample is included, or all samples are the same), and the other is that the height of the tree reaches $log_2\psi$. And the anomaly score is calculated based on the heights for the sample to be isolated in each tree in the forest by equation 3.3.

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{3.3}$$

where x is one instance in training samples, and n is the size of training samples, $h(x) = ln(x) + \xi$ is the length of path for x in a isolation tree, $\xi = 0.5772156649$ is the Euler-Mascheroni constant, $E(h(x))$ is the average length for x in different trees in isolation forest, $c(n) = 2h(n-1) - \frac{2(n-1)}{n}$ is the average length for n samples in binary search tree which is used for normalizing the results[15].
When the anomaly score is close to 1, the average path length is close to 0, it indicates anomalous for that sample as it is easier to be isolated. And when the anomaly score is less than 0.5, the sample can be regarded as normal.

3.3.2 Multivariate normal distribution

Multivariate normal distribution is a density estimation method in machine learning for anomaly detection which calculates the probability of a given data belonging to a training dataset under the assumption that data of all selected features in the training dataset are in Gaussian distribution[1].

The algorithm principle is as follows.

1. Given a normal training set $\{x^{(1)}, ..., x^{(m)}\}$ in n dimensions, fit a model to the data distribution.

$$\begin{aligned} \mu &= \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \\ \Sigma &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T \end{aligned} \tag{3.4}$$

   Where μ is the mean for each feature, and Σ is the covariance.

2. Given a new sample x, compute its probability by

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{3.5}$$

3. Flag an anomaly if $p(x)$ is less than a threshold value ϵ.

The determination of the threshold of probability for anomaly detection by multivariate normal distribution will be explained in section 3.4.

3.3.3 Local outlier factor

Local outlier factor is a density-based or distance-based method for anomaly detection that calculates the local density deviation of a given data point relative to its neighbors[2]. The local outlier factor of a given data point is used for describing how isolated it is from its surrounding k-nearest neighbors, the lower of local outlier factor, the further it is from neighbors, and more possible to be an outlier or anomaly which does not belong to the denser region.
The algorithm principle is as follows[2].

1. Calculate the k-th distance of data point p, that is, the distance of the k-th farthest point from the sample point p, not including p. The common methods of distance are Manhattan distance, Euclidean distance, and Minkowski distance ;

2. Calculate the k-th neighborhood distance of point p, which is all points within the k-th distance of p, including the k-th distance;

3. Calculate the reach-distance. If it is less than the k-th distance, the reach-distance is the k-th distance, if it is greater than the k-th distance, the reach-

distance is the true distance.

The reach-distance for point p and point o is:

$$reach - distance_k(p, o) = max\{k - distance(o), d(p, o)\} \tag{3.6}$$

where d(p,o) is the true distance.

4. Calculate local reachability density of point p which is the reciprocal of the average reach-distance from point p to the points in the k-th neighborhood of point p.

$$lrd_k(p) = \frac{1}{\frac{1}{|Nk(p)|}\sum_{o \in Nk(p)} reach - distance_k(p, o)} \tag{3.7}$$

5. Calculate local outlier factor of point p.

$$LOF_k(p) = \frac{1}{|Nk(p)|}\sum_{o \in Nk(p)} \frac{lrd_k(o)}{lrd_k(p)} = \frac{\sum_{o \in Nk(p)} lrd_k(o)}{|Nk(p)|}\frac{1}{lrd_k(p)} \tag{3.8}$$

For local outlier factor, the value of 1 is the threshold for local outlier factor, samples with lower values are inliers which can be regarded as a denser region, while outliers have a significant higher values and far away from the region[2].

3.3.4 LSTM AutoEncoder

Recurrent neural network (RNN)

RNN is a class of artificial neural networks to analyze sequential data, as the output of the current sub-sequence participates in the calculation of the output of the next sub-sequence, which can be regarded as a passing of memorized information and update. The structure of a standard recurrent neural network is shown in figure 3.3.1, where the neural network on the right is the unfolding structure of that on the left. It can be seen clearly that the nodes between the hidden layers are directionally connected with time changes, and the input of the hidden layer includes not only the output of the input layer but also the output of the hidden layer at the previous moment[3].

Given a sequential dataset $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$, the corresponding hidden state $\boldsymbol{s} = \{s_1, s_2, ..., s_n\}$ and the output sequence $\boldsymbol{o} = \{o_1, o_2, ..., o_n\}$ can be calculated by a

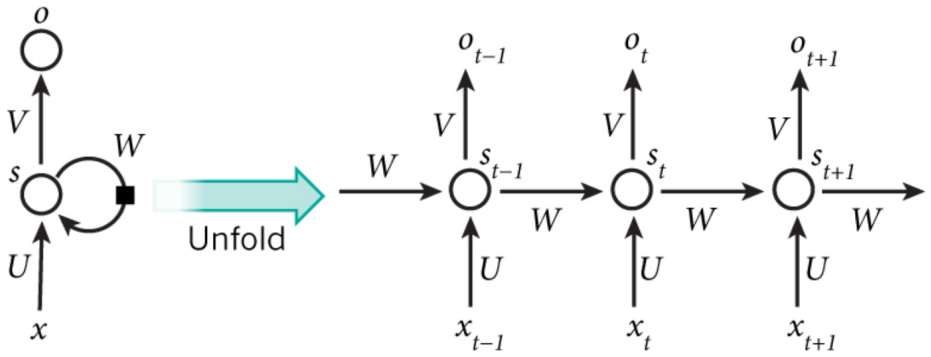

Figure 3.3.1: The structure of a standard recurrent neural network[13]

standard RNN, shown as equation 3.9.

$$\begin{aligned} s_t &= f_a(Ux_t + Ws_{t-1} + b_s) \\ o_t &= Vs_t + b_h \end{aligned} \tag{3.9}$$

Where f_a is the activation function, U is the weight matrix for the input to hidden layers, W is the weight matrix for the recurrent transition between one hidden state to the next, V is the weight matrix for hidden to output transition, b_s and b_h are bias vector for hidden layers and output layers respectively, t is the current time[13].

LSTM

Although RNN can effectively process sequential data, there are still the following two problems: firstly, due to the problems of gradient vanishing and gradient exploding, RNN cannot process time-series data with long-term dependency; then, training the RNN model requires to predetermine the sliding window, but it is difficult to automatically obtain the optimal value of this parameter in practical applications[12]. Therefore, LSTM architecture was introduced to solve the problems above by replacing the RNN hidden unit with LSTM hidden unit[7], shown in figure 3.3.2.
The feed-forward propagation through time in a LSTM unit is shown in equation 3.10.

$$\begin{aligned} \boldsymbol{i}_t &= \boldsymbol{\sigma}(\boldsymbol{W_{xi}}\boldsymbol{x}_t + \boldsymbol{W_{hi}}\boldsymbol{h}_{t-1} + \boldsymbol{W_{ci}}\boldsymbol{c}_{t-1} + \boldsymbol{b_i}) \\ \boldsymbol{f}_t &= \boldsymbol{\sigma}(\boldsymbol{W_{xf}}\boldsymbol{x}_t + \boldsymbol{W_{hf}}\boldsymbol{h}_{t-1} + \boldsymbol{W_{cf}}\boldsymbol{c}_{t-1} + \boldsymbol{b_f}) \\ \boldsymbol{c}_t &= \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t \tanh(\boldsymbol{W_{xc}}\boldsymbol{x}_t + \boldsymbol{W_{hc}}\boldsymbol{h}_{t-1} + \boldsymbol{b_c}) \\ \boldsymbol{o}_t &= \boldsymbol{\sigma}(\boldsymbol{W_{xo}}\boldsymbol{x}_t + \boldsymbol{W_{ho}}\boldsymbol{h}_{t-1} + \boldsymbol{W_{co}}\boldsymbol{c}_{t-1} + \boldsymbol{b_o}) \\ \boldsymbol{h}_t &= \boldsymbol{o}_t \tanh \boldsymbol{c}_t \end{aligned} \tag{3.10}$$

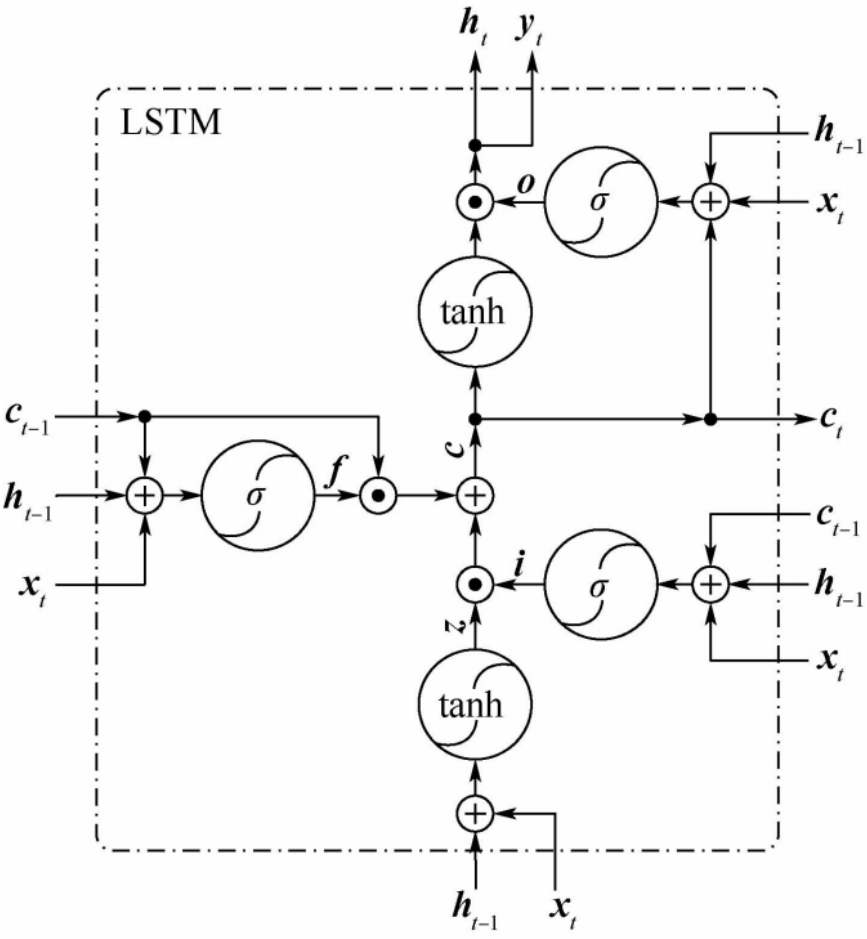

Figure 3.3.2: The structure of a LSTM unit[31]

where $\boldsymbol{i}, \boldsymbol{f}, \boldsymbol{c}, \boldsymbol{o}$ are input gate, forget gate, memory cell and output gate respectively; $\boldsymbol{W}$ is the corresponding weight matrixs and $\boldsymbol{b}$ is the corresponding bias vectors; $\boldsymbol{\sigma}$ is the sigmoid activation function.

AutoEncoder

AutoEncoder is an architecture in a neural network to find the representation of inputs, a technique for representation learning. AutoEncoder includes two parts, encoder which extracts the feature vector $h = f_\theta(x)$ from original input x, and decoder which is a mapping from the compressed feature space to the input space to get a reconstruction $\hat{x} = g_\theta(h)$. The architecture of AutoEncoder is shown in figure 3.3.3.

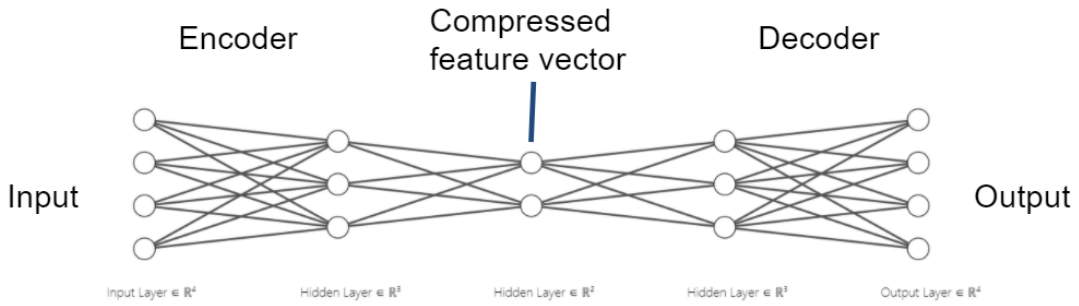

Figure 3.3.3: Encoder-Decoder architecture

By making the reconstruction produced by the decoder as similar as possible to the original data as input, that is, trying to obtain the minimum reconstruction loss $L(x, \hat{x})$ on the training set, the parameter sets θ of the encoder and the decoder is obtained through training at the same time. Good generalization means that a lower reconstruction loss can be obtained in the test set, and the feature vector extracted by the encoder is the representation learned by AutoEncoder.

LSTM AutoEncoder

LSTM AutoEncoder is a combination of LSTM layers and Encoder-Decoder architecture for sequence data, also known as Seq2Seq (Sequence to Sequence).
For a given sequence dataset $\boldsymbol{X} = \{x_1, x_2, ..., x_t\}$ where each sample x_i in $\boldsymbol{x}$ is in D dimensions, a subsequence $x = \{x_1, x_2, ..., x_L\}$ in fixed length L $L \in \{1, ..., t\}$ is fed into LSTM AutoEncoder to encode it, decode it and generate the reconstruction

subsequence $\hat{x} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_L\}$. The training loss in this project is the mean square error between the inputs x and the reconstruction $\hat{x}$, shown in equation 3.11, which is a norm to minimize in the training.

$$Reconstruction\ Loss = MSE(x, \hat{x}) = \frac{1}{L}\sum_{i=1}^{L}(x_i - \hat{x}_i)^2 \tag{3.11}$$

Besides, to evaluate the generalization ability of the model and make a difference from the training loss for avoiding overfitting, the mean absolute error is selected as the anomaly score, shown in equation 3.12.

$$Anomaly\ score = MAE(x, \hat{x}) = \frac{1}{L}\sum_{i=1}^{L}|x_i - \hat{x}_i|^2 \tag{3.12}$$

3.3.5 Hybrid detector

If a well-trained LSTM AutoEncoder has a desired performance on reconstruction, the feature vector in the compressed space can be regarded as the representation of the input data, and the encoder of the model can be used for dimension reduction. The resulting vector can be imported to other machine learning algorithms for anomaly detection, shown in figure3.3.4 as a hybrid detector framework.

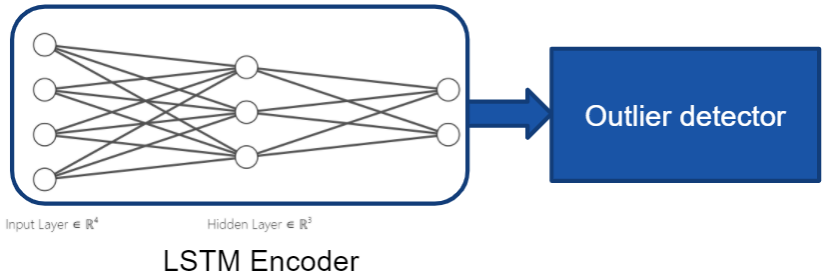

Figure 3.3.4: Hybrid detector

The dimension reduction by LSTM AutoEncoder can capture the nonlinear and temporal-dependent characteristics of the data, compress sequential sample into point sample with little information loss so that the well-defined machine learning algorithms can be used for detecting sequential anomalies, in this hybrid anomaly detection framework the computation costs can be reduced and the performance is guaranteed.

In this project, the outlier detector will be the model with the best performance on point anomaly detection.

3.4 Performance evaluation

Generally speaking, anomaly detection is an unsupervised learning algorithm which means it is impossible to determine whether a sample is anomalous or not just based on the output of the algorithm. Therefore, when developing an anomaly detection system, an efficient method of performance evaluation is necessary for models that do not have a specific decision boundary for the anomaly score. In this project, they are multivariate normal distribution and LSTM AutoEncoder while the decision boundary of isolation forest and that of local outlier factor have already been defined in the algorithms, as mentioned in section 3.3.1 and section 3.3.3 respectively.

3.4.1 Quantitative metrics of performance evaluation

It is necessary to have quantitative metrics to comprehensively evaluate the performance of models and estimate their generalization accuracy.

Confusion matrix

Confusion matrix is a popular method for performance evaluation of classification[26]. In this project, anomaly detection can be regarded as a binary classification problem, anomalous samples are positive samples and normal samples are negative samples. Therefore, a confusion matrix is shown in table 3.4.1.

Table 3.4.1: Confusion matrix

		Actual Values	
		Positive(+)	Negative(-)
Predicted Values	Positive(+)	TP	FP
	Negative(-)	FN	TN

TP is true positve, FN is false negative, FP is false positive, TN is true negative. To have a better performance, higher values in TP and TN while lower values in FP and FN is expected[22].

Precision and recall

As confusion matrix only statistics the number of samples in four categories above, precision and recall are two quantitative metrics in the range of 0 and 1 to evaluate the performance of the model based on the results of confusion matrix. Precision is the percentage of true classification in all predicted positive samples, and recall is the percentage of true classification in all actual positive samples, shown as follows in equation 3.13[22].

$$\begin{aligned} precision_+ &= \frac{TP}{TP + FP} \\ recall_+ &= \frac{TP}{TP + FN} \end{aligned} \tag{3.13}$$

F1 score

F1 score is an advanced quantitative metric that combines the results of precision and recall, shown as follows in equation 3.14[22]. The value of the F1 Score ranges from 0 to 1, where 1 is the best output of the model, and 0 is the worst output of the model.

$$F1\ score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.14}$$

3.4.2 Method of performance evaluation

With a clear quantitative metric F1 score, the method of performance evaluation for non-decision boundary models is:

1. Train the model with labeled training samples
2. Import validation set to the model, get a list of anomaly scores.
3. Set up ϵ candidates within the range of minimum and maximum value of anomaly scores above, and calculate every corresponding F1 score of the validation set.
4. Select the ϵ with the best F1 score as the threshold of anomaly score.
5. Import testing set to the model, evaluate the performance by the selected threshold above.

Besides, the validation set is also valuable for optimization of those models that have clear decision boundaries in the algorithms their selves as the parameters need to be tuned, for example, the number of estimators in Isolation forest and the number of

neighbors in Local outlier factor. With well-tuned parameters, the performance of models will be tested by a testing set.

Chapter 4

Experiments and results

In this chapter, experiments on the implementation of models on vehicle data and the corresponding results are presented. Then, selected from the comparison of performance, the best model will be applied to sequential data.

4.1 Environment set up

The set up of Python environment for implementation is shown in table 4.1.1.

Table 4.1.1: Python environment for implementation

Package	Version
Python	3.7.3
Keras	2.3.1
Matplotlib	3.1.3
Numpy	1.16.4
Pandas	1.0.1
Scikit-learn	0.22.2
Scipy	1.4.1
Tensorflow	2.0.0

4.2 Point anomaly detection

As point anomaly is independent to time, data instance in every timestamp can be transformed into the uniform shape (samples, features) after data preprocessing where features are the corresponding 13 CAN signals for the user function. There are a total of

28 813 samples of normal driving and 9 052 samples of abnormal driving, all labeled with class 0 (or negative) and class 1 (or positive) respectively for better evaluation. To make a balance for two groups of samples, 70% of the samples of normal driving are randomly selected to training models, and the last 8 644 samples of normal driving are mixed with those of abnormal driving for validation and testing, where the splitting ratio for validation set and testing set is 0.5.
Besides, min-max normalization is applied with Isolation forest and Local outlier factor while multivariate normal distribution chooses z-score normalization.

4.2.1 Isolation forest

The model of Isolation forest was built by n base estimators in the ensemble with 256 samples to draw from the training set to train each Isolation tree. The decision boundary for anomaly score is

$$decision\_function = score\_samples - offset. \tag{4.1}$$

where $score\_samples$ is the opposition of the anomaly score, and $offset$ is -0.5, as anomaly score of inliers is close to 0 and that of outliers is close to 1 which has been described in section 3.3.1[21]. Hence, the final decision boundary of anomaly score is 0, positive for inliers and negative for outliers.
Since the algorithm itself has a clear decision boundary, there is no need to determine a threshold again. The number of estimators n in Isolation forest is determined as 29 according the performance on validation set, result shown in figure 4.2.1. The obvious fluctuation of F1 score with the number of estimators increasing confirms that the performance of Isolation forest on the fed data is limited by the performance of base estimators, and the ensemble learning does not help a lot on the improvement of total performance.
And the comprehensive performance of Isolation forest on the testing set is shown in table 4.2.1, where precision and recall, as well as F1 score, are the evaluation metrics, class 0 presents negative samples of normal driving, and class 1 presents positive samples of abnormal driving. The recall of class 1 is very high, at 0.99, while that of class 0 is only 0.42 even lower than random guess (0.5), which means the model can detect most anomalies but also wrongly classify more than half of normal samples into the anomaly.

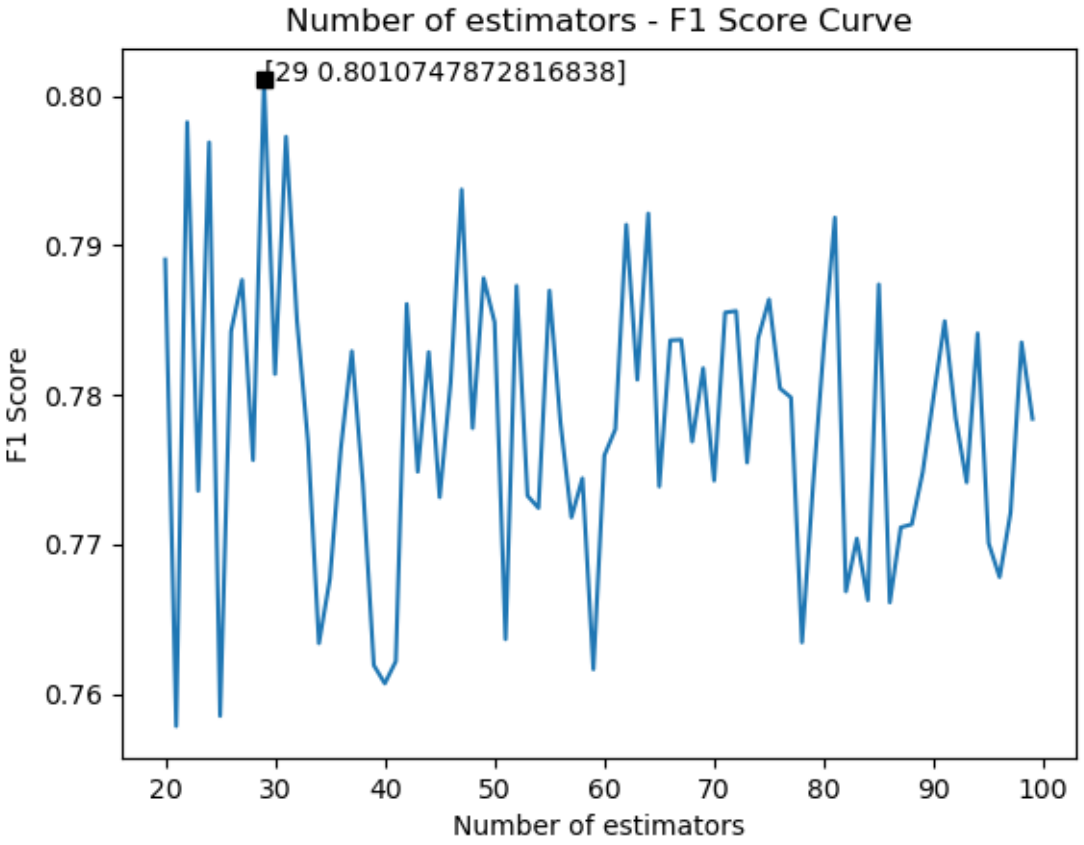

Figure 4.2.1: Number of estimators - F1 Score Curve on the validation set

Table 4.2.1: Classification report of Isolation forest

	precision	recall	F1 score	support
class 0	0.98	0.42	0.59	4310
class 1	0.64	0.99	0.78	4538

The distribution of the anomaly score of the testing set calculated by Isolation forest is shown in figure 4.2.2. Matching the analysis of the classification report in table 4.2.1, large overlapping is in the area below the threshold. The anomaly score of a large number of samples of normal driving is lower than 0 which makes them flagged as anomalies, while for the actual anomaly samples the misclassification happens much less.

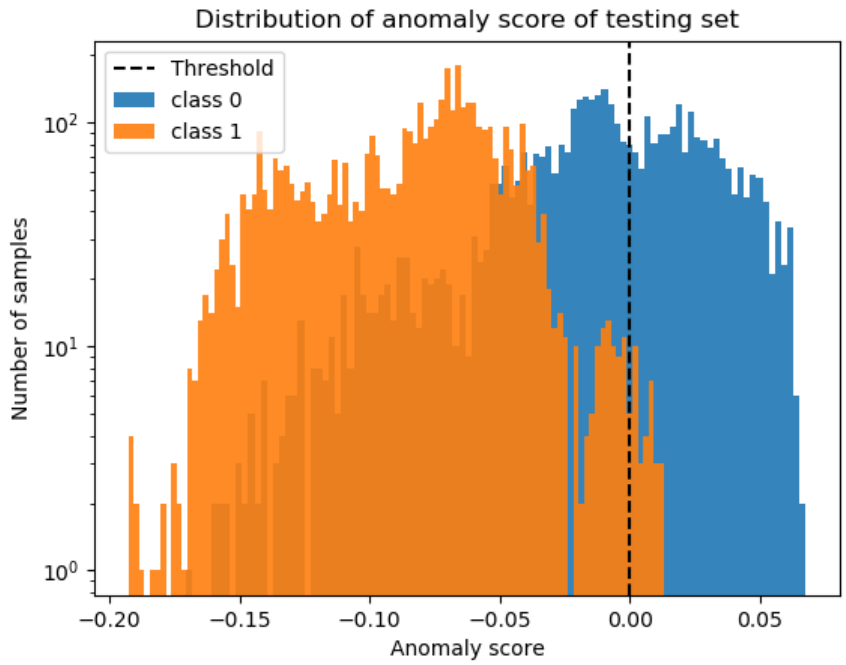

Figure 4.2.2: The distribution of anomaly score of the testing set calculated by Isolation forest

4.2.2 Multivariate normal distribution

The model of multivariate normal distribution is built with training data that has been transformed by z-score normalization. And the anomaly score is to calculate the probability of a sample in the joint probability distribution of the training dataset. Therefore, the lower the probability is, the more possible the sample is to be anomalous, and vice versa.

Since the algorithm itself does not have a clear threshold of probability to determine whether a sample is an anomaly or not, a threshold is determined based on the performance of the model on the validation set, shown in figure4.2.3. The threshold of the anomaly score is the argmax of the F1 Score of the validation set, at $\epsilon =$

$3.710435410458531 * 10^{-05}$. Sample whose anomaly score is lower than the threshold will be flagged as an anomaly.

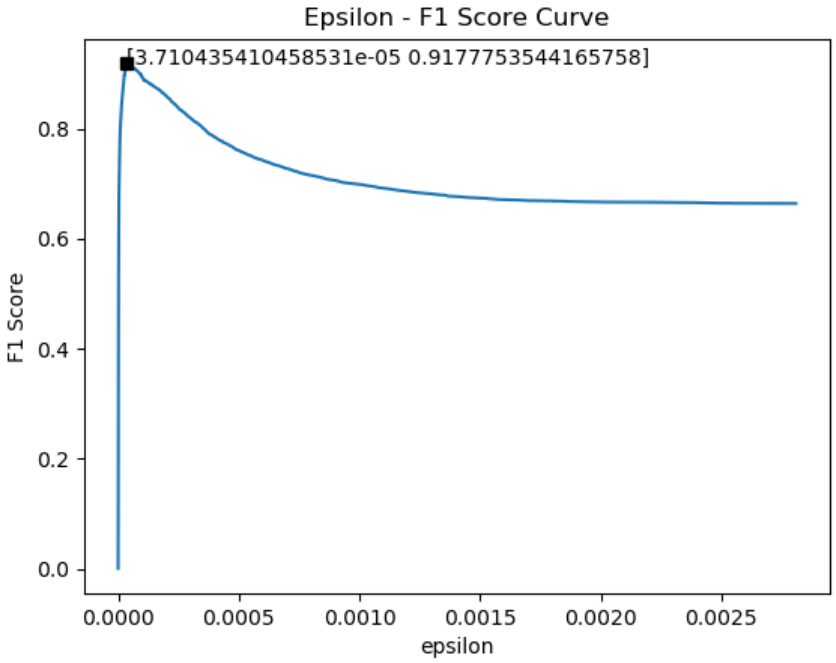

Figure 4.2.3: Threshold determination of multivariate normal distribution by ϵ - F1 score curve on the validation set

With a threshold of anomaly score determined, the comprehensive performance of multivariate normal distribution on the testing set is shown in table 4.2.2. Compared to the results of Isolation forest in table 4.2.1, the recall of class 0 and the precision of class 1 improve significantly, by 0.44 and 0.25 respectively, while the precision of class 0 and the recall of class 1 decrease slightly, which means that the model sacrifices a small part of the ability of anomaly detection but can recognize normal samples better with less misclassification of them.

Table 4.2.2: Classification report of multivariate normal distribution

	precision	recall	F1 score	support
class 0	0.95	0.87	0.91	4305
class 1	0.89	0.96	0.93	4543

The statistic of the anomaly score of the testing set calculated by multivariate normal distribution is shown in figure 4.2.4. When the anomaly score is analyzed it can be seen clearly that the phenomenon of the wrong classification has improved a lot in the normal case with most of the anomaly scores are over the threshold, while that in the

abnormal case does not change a lot.

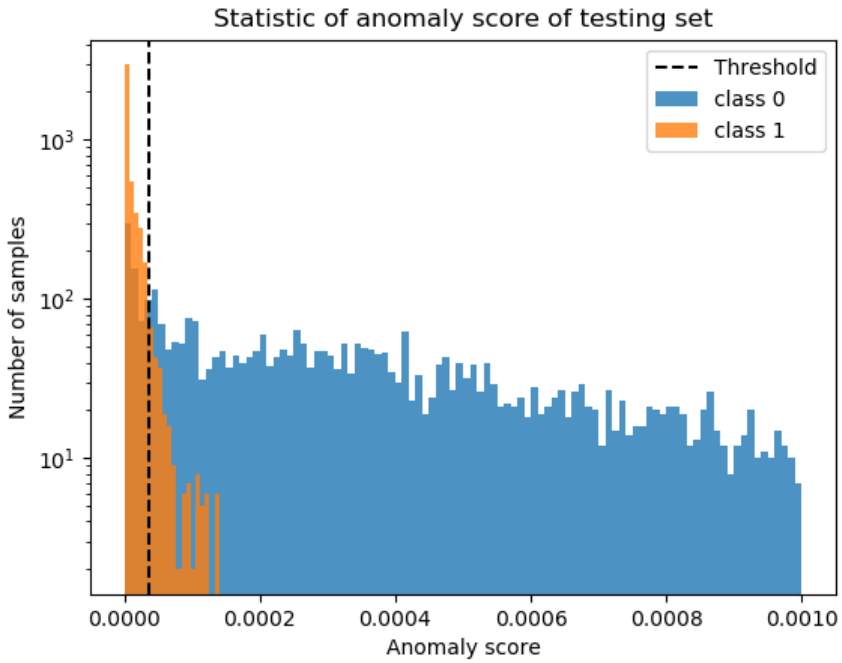

Figure 4.2.4: The distribution of anomaly score of testing set calculated by multivariate normal distribution

4.2.3 Local outlier factor

For the model of Local outlier factor, the Minkowski distance is selected. The decision boundary of the anomaly score is defined by equation 4.2.

$$decision\_function = score\_samples - offset \tag{4.2}$$

where $score\_samples$ is the opposition of Local outlier factor, and offset is -1.5. This shift make a zero threshold for anomaly score, positive for inliers and negative for outliers[21].

The number of neighbors in Local outlier factor is selected as 14 according to the number of neighbors - F1 score curve shown in figure 4.2.5, where 14 is the argmax of F1 score on validation set. It can be seen clearly that when the number of neighbors is larger than 40, there is an obvious decrease of F1 score, which probably comes from underfitting of the model.

The final performance is evaluated by the testing set, results shown in table 4.2.3.

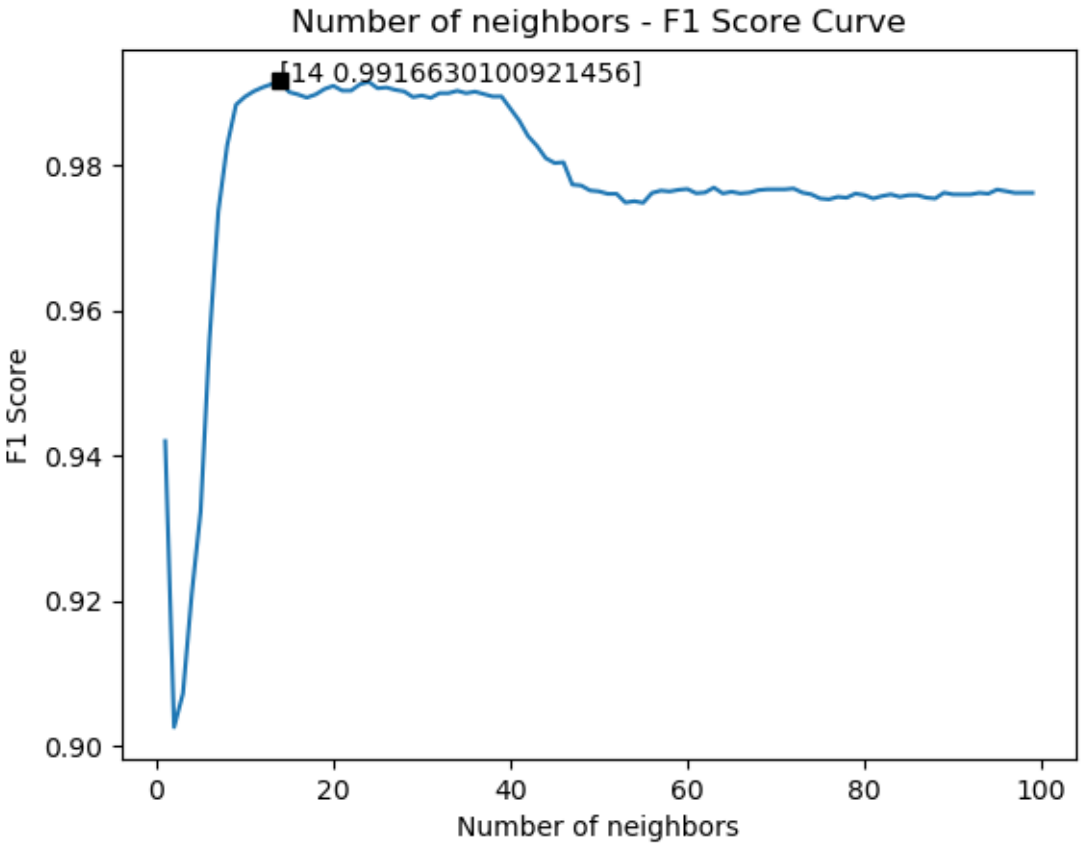

Figure 4.2.5: Number of neighbors - F1 Score Curve on the validation set

Better than multivariate normal distribution, the performance of Local outlier factor has an enhancement on all aspects when compared to that of Isolation forest. The precision of class 0 and the recall of class 1 are up to 1, while the recall of class 0 and the precision of class 1 are 0.98 and 0.99 respectively, which means all the anomalies are detected successfully while there are a few normal samples flagged as anomalies.

Table 4.2.3: Classification report of Local outlier factor

	precision	recall	F1 score	support
class 0	1.00	0.98	0.99	4316
class 1	0.99	1.00	0.99	4532

Figure 4.2.6 visualize the distribution of anomaly score of the testing set calculated by Local outlier factor which is corresponding to the analysis of classification report in table 4.2.3. The anomaly score of all samples of abnormal driving is negative as outliers, the lower, the more abnormal, shown in the orange area in the figure. By contrast, the anomaly score of dozens of samples of normal driving is also negative while most of the samples are positive, which can be regarded as misclassification as they should have been predicted as normality, shown in the blue area in the figure.

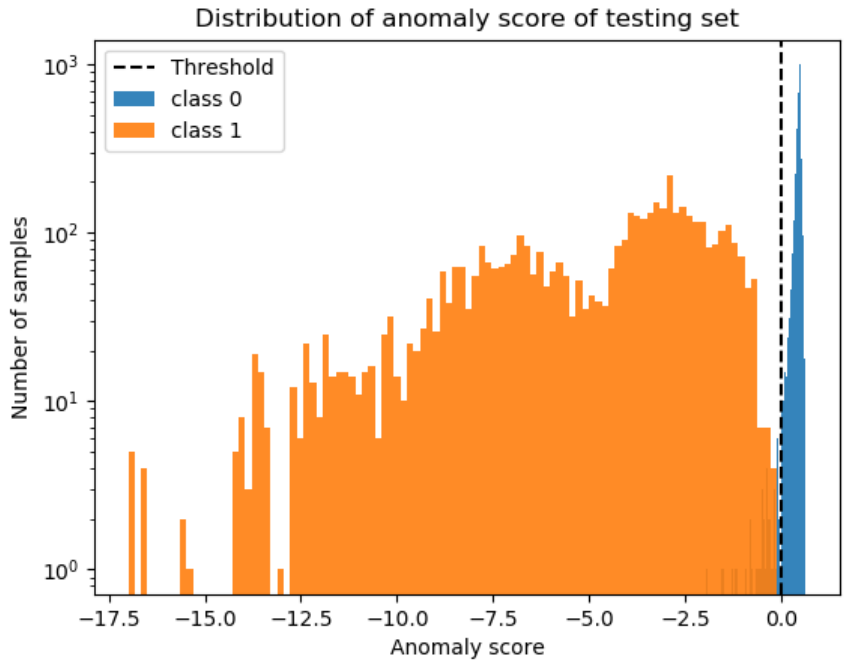

Figure 4.2.6: The distribution of anomaly score of the testing set calculated by Local outlier factor

4.2.4 Performance comparison of point anomaly detection

The comparison of the performance of three point anomaly detection algorithms are shown in figure 4.2.7, where 'MND' represents the model of multivariate normal distribution, and the performance is evaluated by the F1 score.

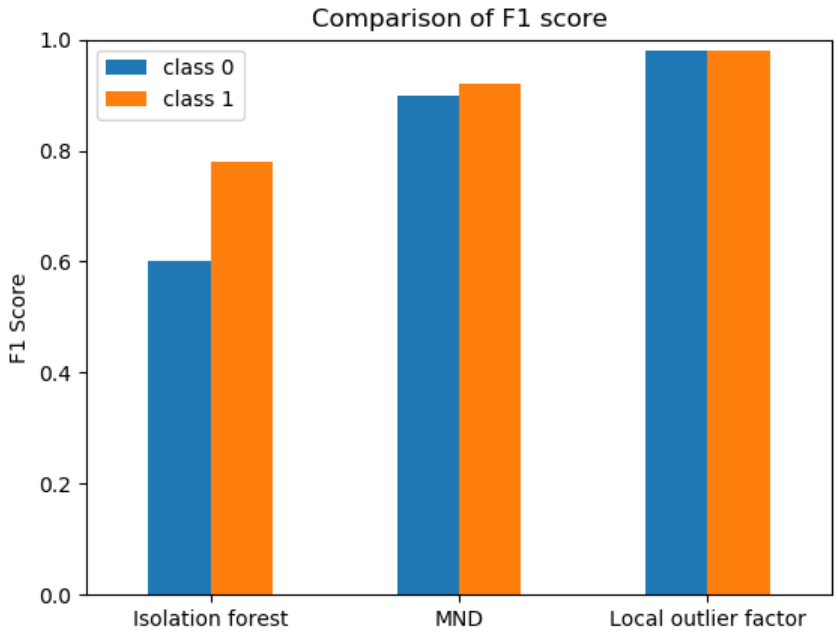

Figure 4.2.7: Comparison of F1 score of three outlier detection algorithms

The hyperparameters for three different point anomaly detection models are tuned separately by the validation set and the generalization abilities are assessed by the testing set. Under the assumption that data in the validation set and the testing set are in a similar distribution, it is surprising for a point anomaly detection algorithm to reach such a good performance on streaming vehicle data with the F1 score of both classes are 0.99, despite there is some false detection of normal samples. Therefore, with the best f1 score among the three outlier detection algorithms, the potential of Local outlier factor will have further experimented within the combination of tensor compression in subsection 4.3.2.

4.3 Temporal dependent anomaly detection

Although contextual anomaly and collective anomaly have different definitions, discussed in section 2.1, both of them show temporal dependence as they usually

appear in some subsequences of time series data. Therefore, to detect temporal dependent anomalies, samples are built in the tensor shape of (samples, timestamps, features) with the sliding window $n = 1$ and the fixed length $L = 4$ file by file due to the time discontinuation among CAN log files. There are a total of 28 781 samples of normal driving and 9 040 samples of abnormal driving, labeled with class 0 (or negative) and class 1 (or positive) respectively for further evaluation, where 70% of the samples of normal driving are randomly selected to train the neural network while the others are used for validation and testing. Additionally, min-max normalization is applied in data preprocessing to speed up the coverage of training.

4.3.1 LSTM AutoEncoder

The LSTM AutoEncoder is built as described in section 3.3.4, within the architecture shown in table 4.3.1. With batch size determined as 32, a tensor in the shape of (32, 4, 13) which includes 4 observations for each of 13 signals for each of 32 samples is fed into an encoder to get the compressed feature space, follower by a decoder to construct outputs in the same shape of inputs. In this neural network, a $softsign$ activation function [23] is applied to each LSTM layer, and a linear activation function is applied to the output layer.

Table 4.3.1: Network architecture of LSTM AutoEncoder

	Layer(type)	Output Shape
Encoder	input_1:Inputlayer	(None, 4, 13)
	lstm_1: LSTM	(None, 4, 64)
	lstm_2: LSTM	(None, 4)
Decoder	repeat_vector_1: RepeatVector	(None, 4, 4)
	lstm_3: LSTM	(None, 4, 4)
	lstm_4: LSTM	(None, 4, 64)
	time_distributed_1(dense_1):TimeDistributed(Dense)	(None, 4, 13)

The model is trained with adam optimizer to minimize the mean square error between inputs and reconstructed outputs[17]. Besides, early stopping is also applied for avoiding overfitting to the training set[17].

In the training of the model, there is a certain percentage of training samples randomly selected in every epoch as the validation samples to monitor the trend of model loss which do not attend the training, in this project the percentage is determined as 10%. The model loss of training is shown in figure 4.3.1, the training terminates at the 45-th

epoch with no more improvement on the val_loss.

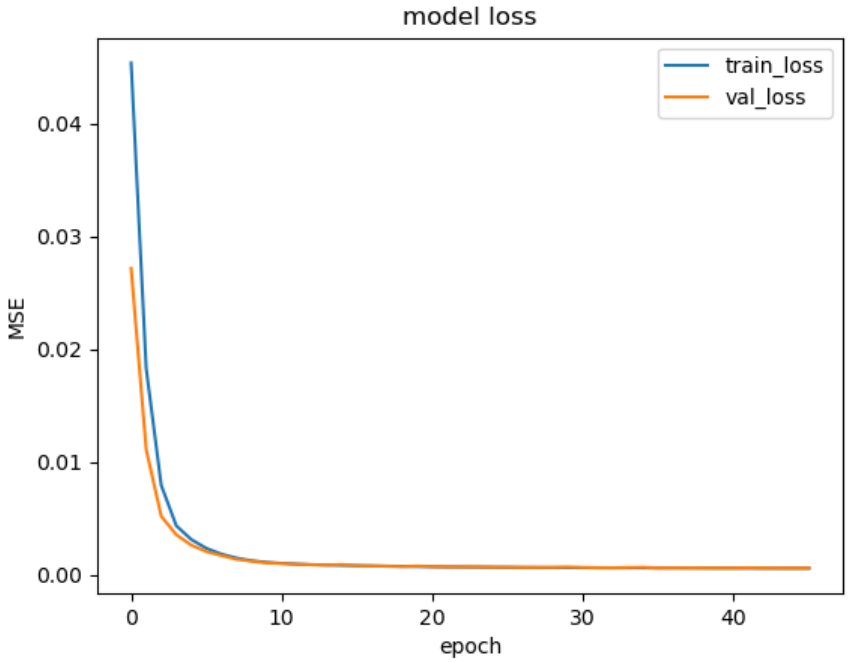

Figure 4.3.1: Model loss of the training of LSTM AutoEncoder

The anomaly score of LSTM AutoEncoder is defined as the mean absolute error between the input and the reconstruction. If a sample get a high anomaly score, it means the sample does not match the pattern of normal driving so it is possible to be flagged as an anomaly.

Since LSTM AutoEncoder itself is just an unsupervised learning algorithm to reconstruct signals, a threshold for anomaly detection needs to be determined by analyzing its performance on the validation set, as described in section 3.4. The result of the threshold determination for LSTM AutoEncoder is shown in figure 4.3.2, where the threshold is the argmax of the F1 score of the validation set, at $\epsilon = 0.03732856855521533$. Sample whose anomaly score is higher than the threshold will be flagged as an anomaly.

Then the generalization capacity of LSTM AutoEncoder is analyzed based on its performance on the testing set which contains 4 388 negative samples (class 0) and 4 450 positive samples (class 1), results shown in table 4.3.2. The performance of anomaly detection on vehicle data is improved when the model takes the factor of time into account, as the precision and the recall of both classes are very close to 1.

Figure 4.3.3 is the statistic results of the anomaly score of the testing set calculated by LSTM AutoEncoder. It can be seen the anomaly score of most negative samples in

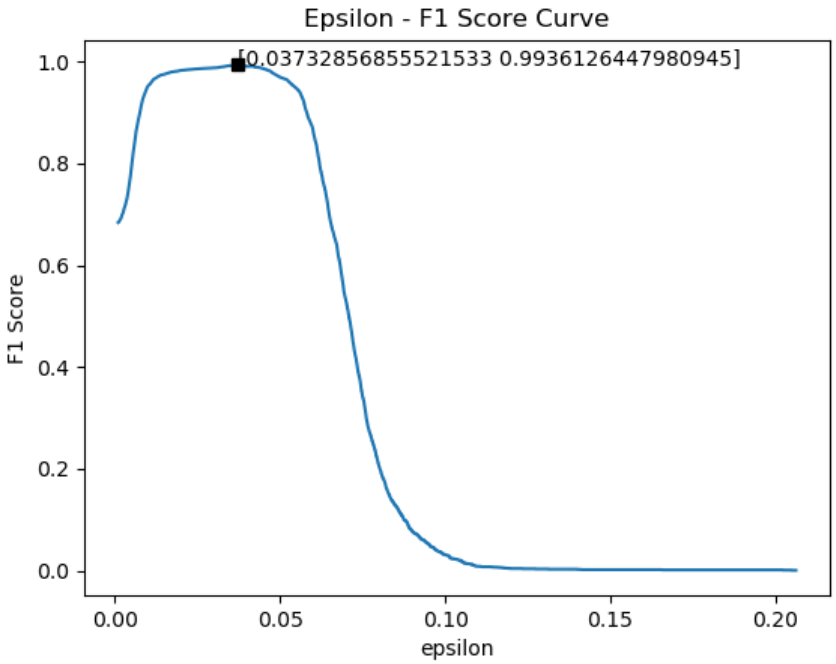

Figure 4.3.2: Threshold determination of LSTM AutoEncoder by ϵ-F1 score curve on the validation set

Table 4.3.2: Classification report of LSTM AutoEncoder

	precision	recall	F1 score	support
class 0	1.00	0.99	0.99	4388
class 1	0.99	1.00	0.99	4450

class 0 are correctly below the threshold, while several samples of class 0 are slightly over the threshold which contributes to the total 1% false positive rate as the precision of the classification of class 1 is 0.99. In the meanwhile, the anomaly score of almost all samples in the abnormal case is over the threshold. Very few overlapping areas of class 0 and class 1 around the threshold shows the excellent performance of LSTM AutoEncoder on anomaly detection.

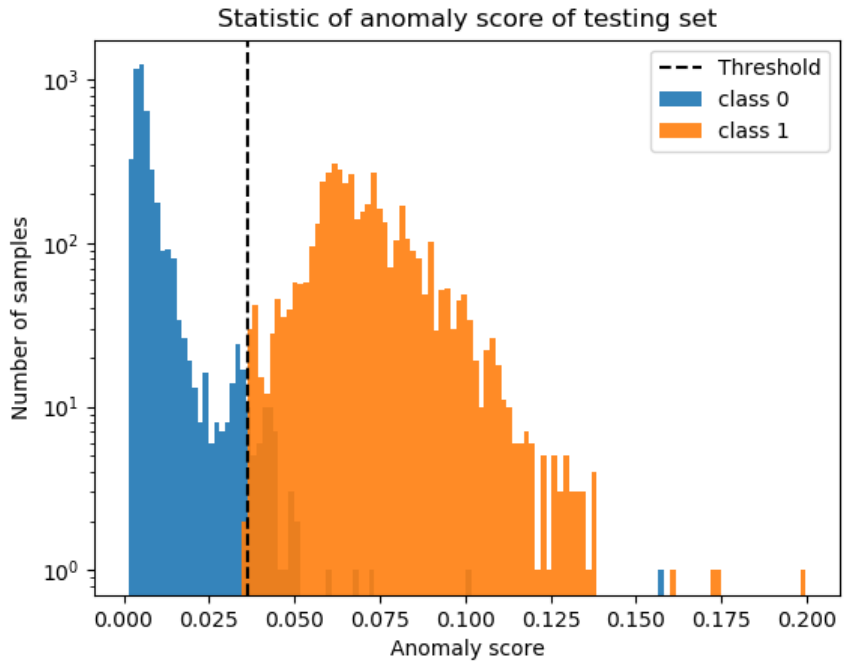

Figure 4.3.3: The distribution of anomaly score of testing set calculated by LSTM AutoEncoder

4.3.2 Hybrid detector

In this project, the hybrid detector is defined as a classical outlier detection algorithm embedded with LSTM Encoder, the framework is described in section 3.3.5.
The LSTM Encoder is part of the architecture in LSTM AutoEncoder, shown in table 4.3.1, to capture the latent valuable information by compressing tensor into lower-dimension feature space. Then, compared from the performance of three different outlier detection algorithms using vehicle data in figure 4.2.7, the model of Local outlier factor is selected to build the hybrid detector whose inputs are those outputs in the shape of (samples, 4 features) from LSTM Encoder. Besides, the decision boundary of anomaly score is still zero, positive for inliers and negative for outliers, described in

section 4.2.3.

The number of neighbors for hybrid detector is determined as 7 according to the results of the number of neighbors - F1 score curve shown in figure 4.3.4, which is the optimal point for the trade off of bias and variance.

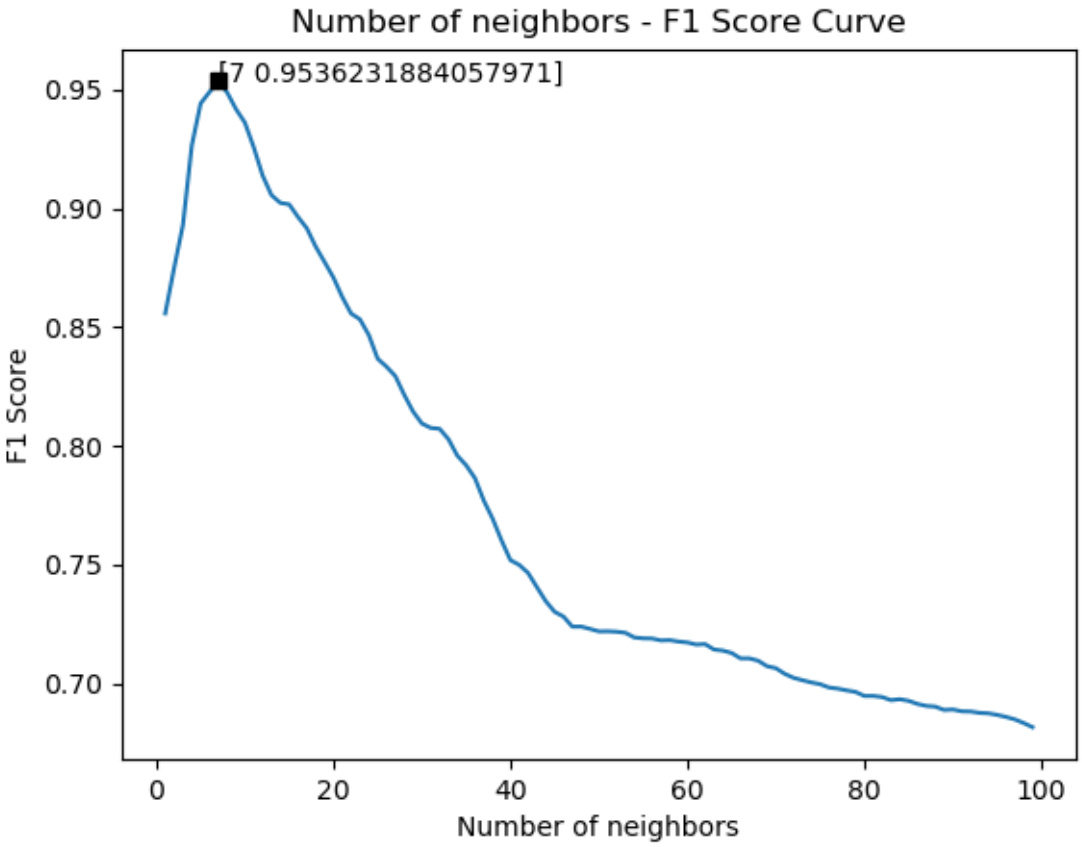

Figure 4.3.4: Number of neighbors - F1 Score Curve on the validation set

The performance of anomaly detection of the hybrid detector on the testing set which contains 4 275 negative samples from normal driving and 4 562 positive samples from abnormal driving is shown in table 4.3.3. The precision and the recall of both classes are unexpectedly lower than either that of Local outlier factor or that of LSTM AutoEncoder, comparison is shown in figure 4.3.5.

Table 4.3.3: Classification report of hybrid detector

	precision	recall	F1 score	support
class 0	0.96	0.96	0.96	4275
class 1	0.96	0.96	0.96	4562

The performance degradation may come from the lossy compression of data from input

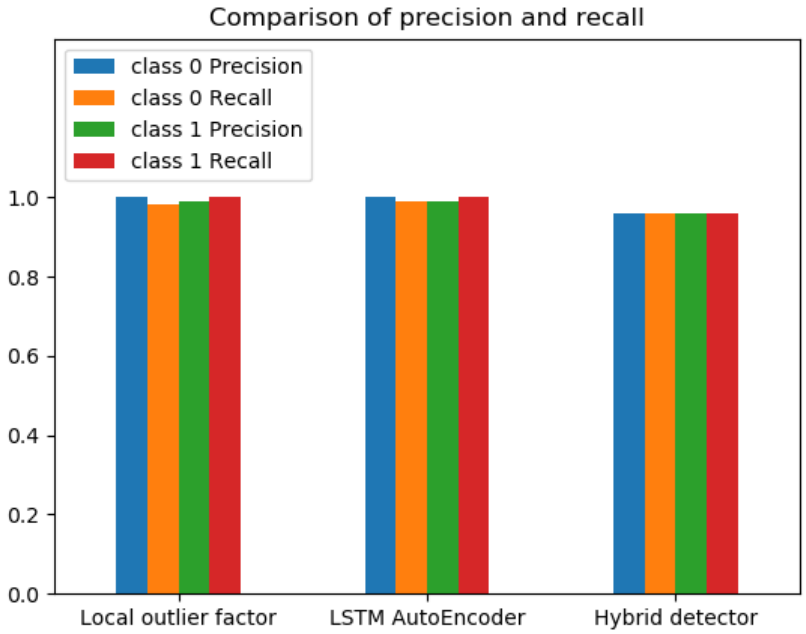

Figure 4.3.5: Comparison of precision and recall of Local outlier factor, LSTM AutoEncoder and hybrid detector

tensor to latent representation by LSTM Encoder. Insufficient information on the nature of the data makes the model hard to capture the hidden pattern in the training dataset, which leads to a higher rate of misclassification between samples of normal behaviors and anomalies.

Figure 4.3.6 shows the distribution of anomaly score of the testing set calculated by hybrid detector, similar with the result of multivariate normal distribution, there are different numbers of samples wrongly classified on both sides of the threshold while both Local outlier factor and LSTM AutoEncoder have strong detection ability of anomalies and keep low false detection on samples of normal behaviors.

4.4 The application of anomaly detection

In this section, the results of the application of the model with the best performance of anomaly detection are illustrated.

Shown in figure 4.3.5, the performance of LSTM AutoEncoder on testing data is slightly better than Local outlier factor, only 0.01 more on the recall rate of class 0, even so, which still proves it can classify better on normal samples. Therefore, in this project, the final model for anomaly detection is selected as LSTM AutoEncoder. With a well-

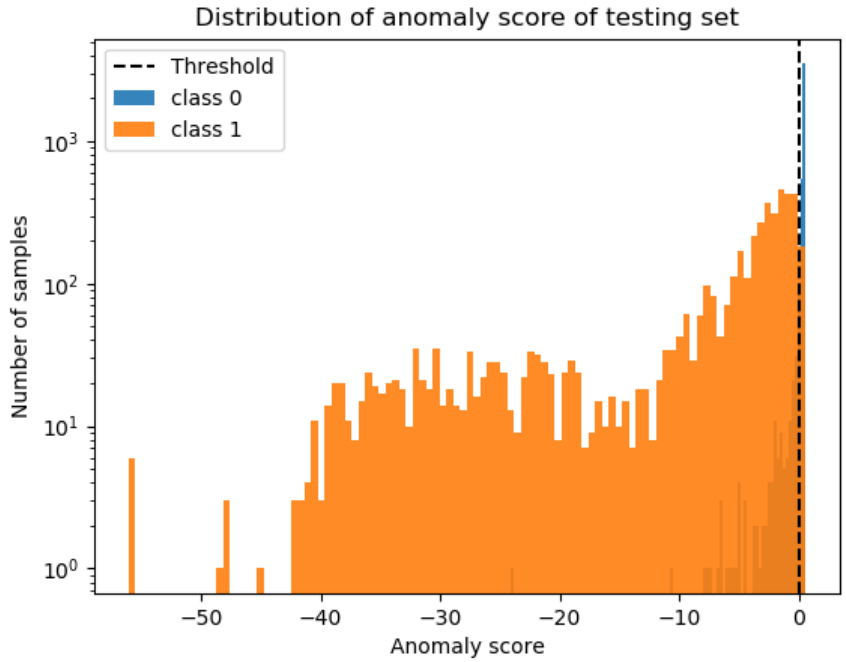

Figure 4.3.6: The distribution of anomaly score of the testing set calculated by hybrid detector

trained model, the process of anomaly detection from raw CAN log file to result is shown as follows.

1. Load model architecture and model weights.
2. Load parameters, including minimum values and maximum values of each feature in the training data for min-max normalization, the fixed length of subsequence as well as the threshold of anomaly score.
3. Import raw CAN log file.
4. Extract signals from CAN message, resample and convert to dataframe, normalize them, and reshape into tensor (samples, timestamps, features).
5. Import testing samples into the model, calculate the anomaly score, and predict.

The results of the application of LSTM AutoEncoder on two raw CAN log files are shown in figure 4.4.1 and figure 4.4.2, from normal driving and abnormal driving respectively. In this case, the anomaly score of all samples of abnormal driving is over the threshold and flagged as anomalies successfully. In the meanwhile, a few samples of normal driving are wrongly detected as anomalies when the anomaly score of other samples are far away from the threshold. Since the figures of results below show the anomaly

score of data over time, the plug-ins in the curve in the figure 4.4.1 are those instances of high anomaly scores while their neighbors in the time series show normality with relatively lower anomaly scores.

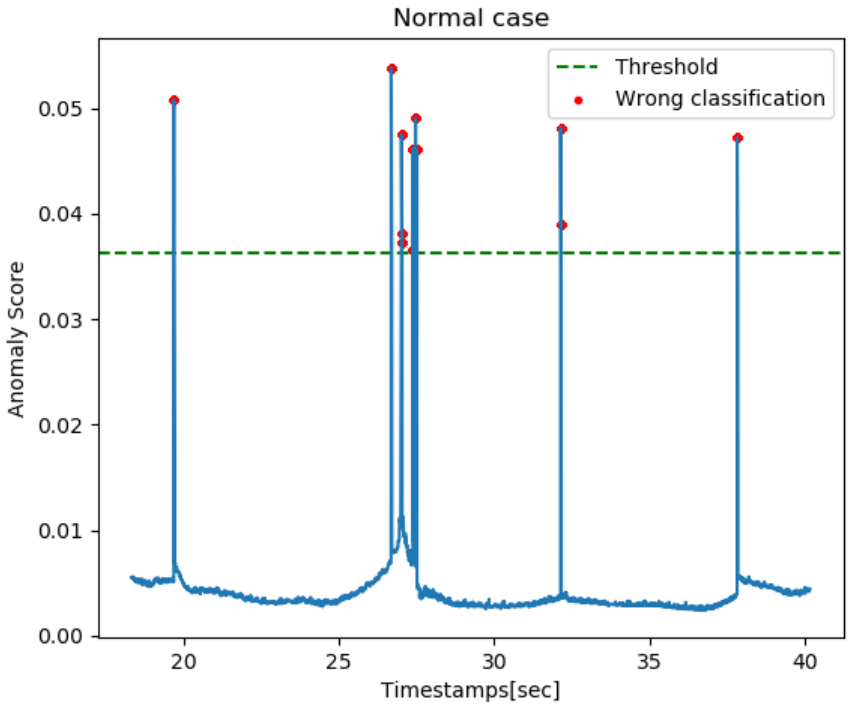

Figure 4.4.1: Results of anomaly score of data of normal driving calculated by LSTM AutoEncoder

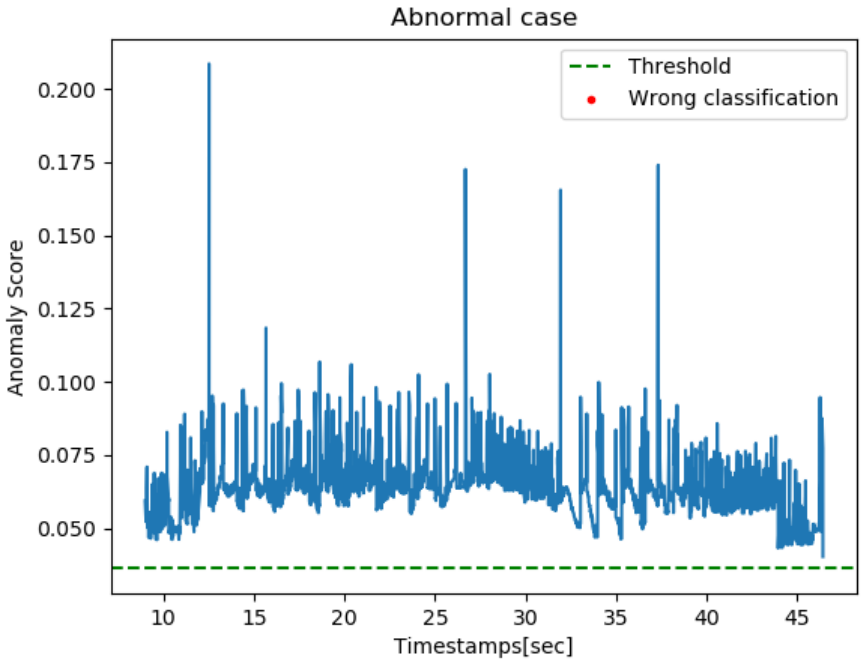

Figure 4.4.2: Results of anomaly score of data of abnormal driving calculated by LSTM AutoEncoder

Chapter 5

Conclusions

This project motivated the need for advanced analysis technique on anomaly detection of user functions as focusing on the increasing difficulty of troubleshooting on vehicle systems caused by the evolution of electrification and autonomous driving. Therefore, different approaches of anomaly detection were experimented by a specific case of user functions, engine speed control using accelerator pedal, with CAN data obtained from software-in-loop simulation which were clearly labelled of normal driving and abnormal driving. By the comparison of the performance of models on testing dataset which contains positive samples and negative samples in the almost same proportion, finally LSTM AutoEncoder was selected to detect the sequence data anomalies logged from CAN bus. The results show promising performance of detection on this kind of functional failure, and prove the feasibility to develop such a technique for automated anomaly detection on vehicle systems.

5.1 Discussion

In the beginning of this project, some research questions were addressed in section 1.4, which will be answered in this section according to the results of case study on user function– engine speed control using accelerator pedal in chapter 4.

- **What kind of model can detect anomalous behaviours?**
  The research object in this project is user-function level failures which might be any of the three types of anomaly. To detect them, approaches of point anomaly detection and temporal dependent anomaly detection are implemented

and experimented respectively.

For point anomaly detection, three model candidates have experimented, isolation forest, multivariate normal distribution, and local outlier factor. The result of the comparison of the performance of these three candidates shown in section 4.2.4 has proved that local outlier factor is the best choice to be implemented on the fed streaming vehicle data which has the best potential on point anomaly detection with the least false positive rate on the wrong classification of normal samples.

For temporal dependent anomaly detection, two models are implemented, LSTM AutoEncoder, and hybrid detector which is a combination of LSTM Encoder as dimension reduction part and local outlier factor which analyzes the data in compressed feature space. LSTM AutoEncoder, in the architecture of Encoder-Decoder, gets a desired good results on reconstructing inputs of normal samples and distinguish them from those of abnormal samples by a much higher anomaly score, the statistic of anomaly score of testing samples shown in figure 4.3.3. However, the performance of the hybrid detector is unexpectedly worse than that of separate implementation of any sub-models in its framework, either temporal dependent anomaly detection by LSTM AutoEncoder or point anomaly detection by local outlier factor, comparison shown in figure 4.3.5. Generally speaking, dimension reduction is a technique used for speeding up the training of the model and visualizing the feature space for those data in high dimensions. In this project, there are only 13 features of the user function selected for training the model, which has not too much burden on training speed or system memory. Extra dimension reduction will lead to information loss in the feature space compression by LSTM Encoder, which makes the model harder to capture the hidden pattern in the data, that is the reason for performance degradation.

Therefore, finally, the model of LSTM AutoEncoder is selected as a core of data analysis to develop the anomaly detection technique, the results of its application on streaming data in figure 4.4.1 and figure 4.4.2 shows the promising performance as detecting most anomalies with little wrong classification on normal samples.

- **Which model have the best potential to be applied to different user functions?**

  LSTM AutoEncoder has proved its potential to be applied to anomaly detection

by reconstructing the input sequence and monitoring the reconstruction error, the results of its application on two raw CAN log files shown in figure 4.4.1 and figure 4.4.2. As a type of unsupervised learning model, it provides the freedom of the definition of the threshold of anomaly score which can be adjusted depending on the needs. Moreover, within the acceptable range of system memory and training speed, it does not need too much preprocessing and analysis of the feature of data before training. To be applied to different user functions, it would be an individuation process for the model as the hyperparameters need to be optimized by cases, such as the depth of the neural network, the number of hidden units of different layers, the activation function, and the fixed length for subsequence.

Besides, dimension reduction is still worth a try for those high dimensional data, such as those user functions with hundreds of CAN signals, but maybe using other techniques such as PCA[28] or Kernel PCA[18] and embedding them as a feature analysis part before LSTM AutoEncoder.

Besides, there are some drawbacks and limitations to the work. Firstly, the data used in this project comes from software-in-loop simulation, which can be regarded as clean synthetic data with no noise. Lack of training and testing with enough experimental data from the real world, the model cannot be applied to vehicle systems directly to see its performance. Instead, the results only show the feasibility of developing such an AI technique to detect failures in user function. Even so, the data of abnormal driving still cannot cover all forms of failures, though the model is trained to learn the pattern of normal driving, which will introduce some deviation in the threshold determination and influence the final performance.

Secondly, the non-uniform update sampling frequencies of different CAN messages might bias the model to those training data samples which are sampled multi times, which makes the model overfitting.

Further, the phenomenon of memory runs out appeared in the training of neural networks with hundreds of thousands of data when the length of the subsequence sample increased to 8 and more, which limited the optimization of hyperparameters of the model. Therefore the further application of GPU parallel computing and cloud computing might bring fresh and surprising improvement for the model performance.

5.2 Future work

The case study of anomaly detection in this project is the key milestone to establish a large-scale, automated anomaly detection system of monitoring streaming data on Scania vehicles. Future work will focus on the improvement of the performance of the off-board model, as well as the deployment of the onboard model.

Firstly, the most important point is to train and test the performance of the model by feeding CAN data logged from the real world into it, which can be realized by starting with data from the hardware-in-loop simulation, and then data logged from field tests of vehicles. If using the same signals of the user function, and the same architecture of the neural network which is shown in table 4.3.1, the training work can start with the weights saved before, which can help save unnecessary time and speed up the training process. Otherwise, the optimization of hyperparameters of neural networks should be conducted by Bayesian Optimization[25] or other optimization methods to look for better performance.

Popular deep generative models like Variational Autoencoders[20] and Generative Adversarial Networks[14] are also worth experimented with combined with LSTM layers to see their performance of anomaly detection on vehicle data. Compared to LSTM AutoEncoder which is selected in this project, they are more powerful to find the hidden pattern in normal data, with enough data fed.

Furthermore, an approach of a dynamic threshold of anomaly score could be worth explored for improving the robustness of the system, which can be realized by discovering the law between anomaly score of samples and their representation in compressed feature space[20].

Regarding model deployment in real-time, data acquirement from CAN bus and preprocessing before imported into the model should be considered carefully, so should the time and memory needed for running the model, as it should be online anomaly detection with not too much burden to the system.

Bibliography

[1] Akouemo, Hermine N and Povinelli, Richard J. "Probabilistic anomaly detection in natural gas time series data". In: *International Journal of Forecasting* 32.3 (2016), pp. 948–956.

[2] Breunig, Markus M, Kriegel, Hans-Peter, Ng, Raymond T, and Sander, Jörg. "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[3] Chalapathy, Raghavendra and Chawla, Sanjay. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019).

[4] Chandola, Varun, Banerjee, Arindam, and Kumar, Vipin. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[5] Cho, Sung-Bae and Park, Hyuk-Jang. "Efficient anomaly detection by modeling privilege flows using hidden Markov model". In: *computers & security* 22.1 (2003), pp. 45–55.

[6] Fu, Zhouyu, Hu, Weiming, and Tan, Tieniu. "Similarity based vehicle trajectory clustering and anomaly detection". In: *IEEE International Conference on Image Processing 2005*. Vol. 2. Ieee. 2005, pp. II–602.

[7] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT press, 2016.

[8] Görnitz, Nico, Kloft, Marius, Rieck, Konrad, and Brefeld, Ulf. "Toward supervised anomaly detection". In: *Journal of Artificial Intelligence Research* 46 (2013), pp. 235–262.

[9] Hawkins, Douglas M. *Identification of outliers*. Vol. 11. Springer, 1980.

[10] Hundman, Kyle, Constantinou, Valentino, Laporte, Christopher, Colwell, Ian, and Soderstrom, Tom. “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding”. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 387–395.

[11] Justin, D, Concepcion, Ronnie S, Bandala, Argel A, and Dadios, Elmer P. “Performance Comparison of Classification Algorithms for Diagnosing Chronic Kidney Disease”. In: *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*. IEEE, pp. 1–7.

[12] Lechner, Mathias and Hasani, Ramin. “Learning Long-Term Dependencies in Irregularly-Sampled Time Series”. In: *arXiv preprint arXiv:2006.04418* (2020).

[13] LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[14] Li, Dan, Chen, Dacheng, Jin, Baihong, Shi, Lei, Goh, Jonathan, and Ng, See-Kiong. “MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks”. In: *International Conference on Artificial Neural Networks*. Springer. 2019, pp. 703–716.

[15] Liu, Fei Tony, Ting, Kai Ming, and Zhou, Zhi-Hua. “Isolation forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[16] Maia, Rui. “Multivariate temporal data analysis for vessels behavior anomaly detection”. In: *ac. uk/EuroDW2018/papers/eurodw18-Maia. pdf* (2018).

[17] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Yangqing, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on*

Heterogeneous Systems. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[18] Mika, Sebastian, Schölkopf, Bernhard, Smola, Alex J, Müller, Klaus-Robert, Scholz, Matthias, and Rätsch, Gunnar. "Kernel PCA and de-noising in feature spaces". In: *Advances in neural information processing systems*. 1999, pp. 536–542.

[19] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[20] Park, Daehyung, Hoshi, Yuuna, and Kemp, Charles C. "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1544–1551.

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

[22] Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: (2011).

[23] Qiu, Dawei, Liu, Zichen, Zhou, Yiqing, and Shi, Jinglin. "Modified Bi-Directional LSTM Neural Networks for Rolling Bearing Fault Diagnosis". In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[24] Quackenbush, John. "Microarray data normalization and transformation". In: *Nature genetics* 32.4 (2002), pp. 496–501.

[25] Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.

[26] Stehman, Stephen V. "Selecting and interpreting measures of thematic classification accuracy". In: *Remote sensing of Environment* 62.1 (1997), pp. 77–89.

[27] Taylor, Adrian, Leblanc, Sylvain, and Japkowicz, Nathalie. “Anomaly detection in automobile control network data with long short-term memory networks”. In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE. 2016, pp. 130–139.

[28] Verleysen, Michel and François, Damien. “The curse of dimensionality in data mining and time series prediction”. In: *International work-conference on artificial neural networks*. Springer. 2005, pp. 758–770.

[29] Vignisson, Egil. *Anomaly Detection in Streaming Data from a Sensor Network*. 2019.

[30] Wang, Liying, Shi, Lei, Xu, Liancheng, Liu, Peiyu, Zhang, Lindong, and Dong, Yanru. “A Parameter-Free Outlier Detection Algorithm Based on Dataset Optimization Method”. In: *Information* 11.1 (2020), p. 26.

[31] Wang, Xing, Wu, Ji, Liu, Chao, Yang, H, Du, Y, and Niu, W. “Exploring LSTM based recurrent neural network for failure time series prediction”. In: *J. Beijing Univ. Aeronaut. Astronaut* 44.4 (2018), pp. 772–784.

[32] Zhao, Manqi and Saligrama, Venkatesh. “Anomaly detection with score functions based on nearest neighbor graphs”. In: *Advances in neural information processing systems*. 2009, pp. 2250–2258.

[33] Zhu, Xiaojin and Goldberg, Andrew B. “Introduction to semi-supervised learning”. In: *Synthesis lectures on artificial intelligence and machine learning* 3.1 (2009), pp. 1–130.

TRITA -SCI-GRU 2020:337