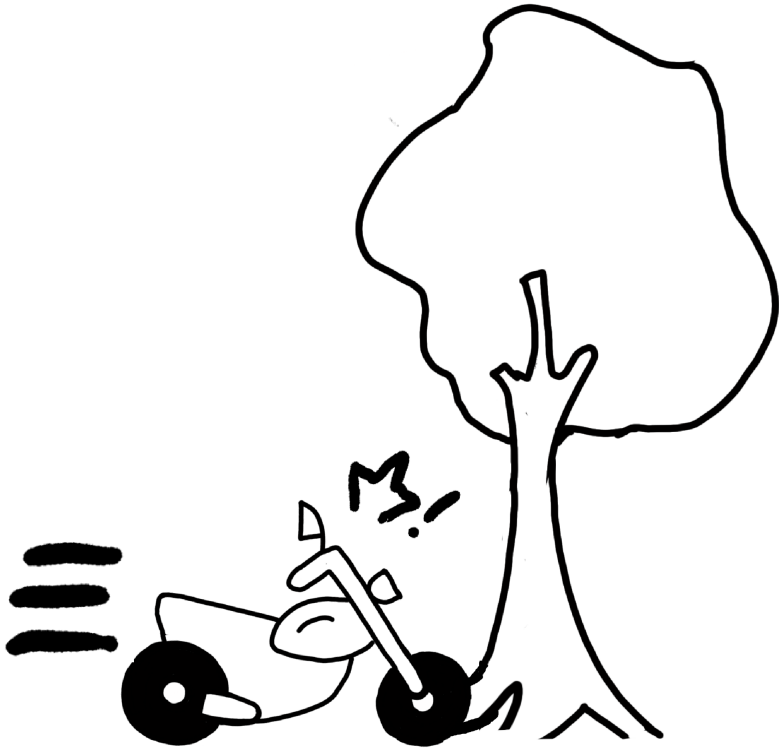
CHALMERS
UNIVERSITY OF TECHNOLOGY



Development of a Crash Detection Algorithm for Motorcycle Drivers using Machine Learning

Master's thesis in Electrical Engineering

LINGYUN DENG

DEPARTMENT OF ELECTRICAL ENGINEERING

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

Master's thesis 2021

Development of a Crash Detection Algorithm for Motorcycle Drivers using Machine Learning

LINGYUN DENG

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Development of a Crash Detection Algorithm for Motorcycle Drivers using Machine Learning
LINGYUN DENG

© LINGYUN DENG, 2021.

Supervisor: Stefan Candefjord, Department of Electrical Engineering, Chalmers University of Technology
Supervisor: Andreas Carlsson, Detecht Technologies AB
Supervisor: Niklas Ohlsson, Detecht Technologies AB
Examiner: Bengt Arne Sjöqvist, Department of Electrical Engineering, Chalmers University of Technology

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

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Development of a Crash Detection Algorithm for Motorcycle Drivers using Machine Learning
Lingyun Deng
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The motorcycle is a popular vehicle that can be used for multiple purposes, such as sport and commuting. It is estimated that only in Europe there are already 23 million motorcycles. However, statistics show that its risk of severe or deadly crashes is over-represented in traffic fatalities, of which one important reason is the lack of support when the motorcyclists are driving alone so that the timely medical treatment is missed to improve the wounded outcome. To tackle this problem, Detecht Technologies AB has developed a smartphone application which can automatically send rescue information to public-safety answering point when the app detects a crash event by measuring the embedded sensors of the smartphones. The aim of this thesis is to improve the motorcycle crash detection algorithm with machine learning, based on the driving data provided by Detecht.

The dataset is composed of 500 normal driving logfiles without crash events and 17 logfiles each containing one crash event. In total there are recordings of about 630 driving hours. A literature review is conducted to study the existing methods of motorcycle crash detection. In addition, several deep learning based temporal anomaly detection methods are researched and candidates methods are compared. Furthermore, an exploratory data analysis is conducted to understand the composition and limitation of the dataset. Then a crash detection algorithm pipeline is proposed based on the findings. The method is the autoencoder based to learn the generic normality features and then to define the anomalous patterns as possible crash events. Several autoencoders are tested and their performance are evaluated on the same dataset, and an optimal model, which is a MobileNetV3-style autoencoder, is standing out by comparing their ability of detecting true crashes while maintaining low false alarm rate. Finally, the motorcycle crash detection results are compared with the given true crashes and normal driving, from the sense of sensitivity and specificity.

It shows that the optimal model can detect all the provided true crash events while raising a false alarm per 1.8 driving hours on average. The thesis analyzes these results and discusses the relationship between crash detection and false alarms arousing. In addition, several suggestions are proposed and analyzed as a future development direction.

Keywords: machine learning, anomaly detection, autoencoders, convolutional neural networks, motorcycle crash detection.

Acknowledgements

I'd like to thank certain people for their invaluable help without personal gain during the thesis work. I would like to thank my supervisors Stefan Candefjord, Andreas Carlsson, Niklas Ohlsson and examiner Bengt Arne Sjöqvist for giving me the opportunity to conduct this thesis at Detecht and Chalmers. I am grateful to Stefan Candefjord for his time, feedback, and genuine kindness. My warm regards to Andreas Carlsson for his generous technical help and precious suggestions. Their professional and patient supervision has greatly impressed me and discussion with them in biweekly meetings and flexible Slack has given me endless ideas and inspirations. Better than a thousand days of diligent study is one day with a great teacher. I also want to express my sincere gratitude to Sabine Reinfeldt, Xuezhi Zeng, and Lena Sommarström, for their effort in helping me find this thesis.

I would like to thank my opponents, Chattarin Wangwittaya and Yuling Zhang, for their contribution of critique. And thanks for all the authors of their previous work, Gabriel Matuszczyk, Rasmum Åberg, Linh Tuan Nguyen, and Saud Ahmad Mian. Without their foundations my thesis cannot go so far. In addition, I would like to thank Chalmers University of Technology for its great courses, activities, resources, and environment. My choice of Chalmers is with no regrets.

Finally, special thanks to my parents supporting me to pursue the full and free development of man, no matter they agree or disagree with my every decision. Thanks to my friends for their continuous encouragement and motivation. Thank my brother Siyuan Wang for drawing the cover image. None mentioned, none forgotten.

Tack!
Thank you!

Lingyun Deng, Gothenburg, June 2021

Contents

List of Figures

List of Tables

Acronyms

AE autoencoder. 10–12
AUC Area under the ROC Curve. 6, 16

CH Challenge. 7
CNN convolutional neural network. 12
CNN-AE convolutional autoencoder. 11, 15

FPR false positive rate. 6, 15, 20, 23

GAN generative adversarial network. 11
GNSS global navigation satellite system. 5
GRU gated recurrent unit. 11

IMU inertial measurement unit. 5

LSTM long short-term memory. 11
LSTM-AE long short-term memory autoencoder. 11

MAP maximum a posteriori. 6
MCC motorcycle crash. 1–7, 9, 20, 45
mobile-AE mobileNetV3-small style autoencoder. 21
MSE mean squared error. 23

NAS neural architecture search. 47
NN neural network. 9, 10, 12, 21
NPU neural processing unit. 41

OCSVM one-class support vector machine. 5, 6, 11

RNN recurrent neural network. 11, 12, 47
RNN-AE RNN autoencoder. 11
ROC receiver operating characteristic. 6, 17

SVDD support vector data description. 11

TPR true positive rate. 6, 23, 27

VGG-AE VGG style autoencoder. 21

1
Introduction

1.1 Background

Motorcycles can be used in a range of versatile purposes, such as sport and commuting. The Federation of European Motorcyclists' Associations (FEMA) estimated that there are 23 million motorcycles in Europe [4]. However, statistics show that the risk of death or injury for motorcycle drivers due to traffic accidents is overrepresented in traffic fatalities [5, 6, 7]. In Europe, there is more than 3644 deaths in road accidents involving riders of motorcycles in 2016, and in Sweden, the statistic is around 40 fatalities yearly [8, 9]. Besides, 34% of powered-two-wheelers casualties were admitted to the hospitals, and they stayed for about 10 days on average [8]. The statistical data of the road traffic accidents reported by the Swedish police is presented in Table 1.1. Compared with car crashes, though the number of car crashes is over ten times than of motorcycle crash (MCC), their severity tells the difference: the percentage of severe and deadly accidents in MCC accounts for near three times than that in car crashes.

Severity	Car	Motorcycle	Moped	Cyclist	Pedestriar	Other	Total
Light	8875	620	883	1998	1017	316	13709
Severe	804	230	110	239	198	64	1645
Deadly	122	28	4	18	25	7	204
Total	9801	878	997	2255	1240	387	15558
Percent of accident per severity level and road user group [%]							
Light	90.55	70.62	88.57	88.60	82.02	81.65	88.12
Severe	8.20	26.20	11.03	10.60	15.97	16.54	10.57
Deadly	1.24	3.19	0.40	0.80	2.02	1.81	1.31

Table 1.1: Statistics accident severity level comparison by different road users in 2020 by the official statistics of Sweden. Calculated from [1].

Analyzing the cause of MCC may help find the distinction. MCC can be divided into two categories, single accidents (involving only one motorcycle) and collisions with another vehicle (such as a car), and as shown in Table 1.2 different accident conditions lead to different physical injury risks, according to FEMA [4]. The injury risk percentages is calculated from an FEMA survey of Dutch motorcyclists, and can be an indicative of the European situation. In total, each category takes up half of the all accidents, and the most likely top 13 accident risks are listed in Table 1.2 that are responsible for more than half of all the MCC.

Accident condition	Physical injury risk
Single accidents	
Emergency stop to avoid accident (mostly with cars)	H
Oil or fuel on road	M
Steering error especially in curves	M
Braking error and blocking brakes	M
Collision	
Car coming sideways not yielding right of way on intersection	H
Car coming onto road from parking area, outlet, gas station etc.	H
Car oncoming from opposite direction and turning left in front of motorcycle	H
Car oncoming from opposite direction driving in wrong lane	H
Car moving in same direction changing lanes (overtaking, making left/right turn, parking)	M
Car moving in same direction hitting motorcycle from behind stopped for traffic light, yielding etc.	M
Car moving in same direction changing lanes in traffic jam	M
Car moving in same direction hitting riding motorcycle from behind	M
Motorcycle not keeping enough distance to car in front moving in same direction	L

Table 1.2: Top 13 accident risks with its physical injury indicator in alphanumerical order. High(H): Over half chance of physical injury. Medium (M): more than one-quarter chance of physical injury. Low(L): less than one-quarter chance of physical injury. Adapted from [4].

The graph of the number of fatalities in road accidents of car crashes and MCC in Sweden is presented in Fig 1.1, and the graph is drawn based on [1]. It shows that the fatality rate of car crashes is generally decreasing, particularly in recent decades, but that of MCC is not decreasing. The reason might be due to that, the shell of the car can act as a protective structure, combining with more paramount safety measures are introduced to car year by year, while the design of motorcycle makes its driver more vulnerable [7, 10] hence effective protections for motorcycle are in demand. In addition, for Table 1.1, it is worth mentioning that while the death of car crashes contains both drivers and passengers, the death in MCC contains only the drivers in 2020, which suggests that protective safety measures for motorcyclists are particularly needed for the case when they are driving alone [5, 11].

To tackle this problem and protect the motorcyclists in crashes, several approaches are proposed and deployed. They can be classified into passive and active safety strategies. For example, wearing a helmet in the proper manner can effectively reduce the risk of death and severe injury, though it is found that less than half of all countries have legislated specific standards for helmets [12]. Besides, there are both chest airbags on the motorcycle and protective clothing embedded with airbags to reduce the serious impact-related injuries [13]. In addition to passive equipment, similar to the eCall systems which are mandatorily deployed in automotive, eCall for motorcycles are initiated. eCall means that a call is made to an emergency service in order to rescue the people. Imagine when a severe MCC happens, the automatic crash detection system can detect the crash, locate the driver, and notifying relevant information to the emergency service. In this case, the drivers are more likely to receive medical treatment, without counting on the possible passersby. In short, all these above-mentioned efforts require a reliable and effective MCC detection method. Therefore, real-time motorcycle crash detection, from one side is needed

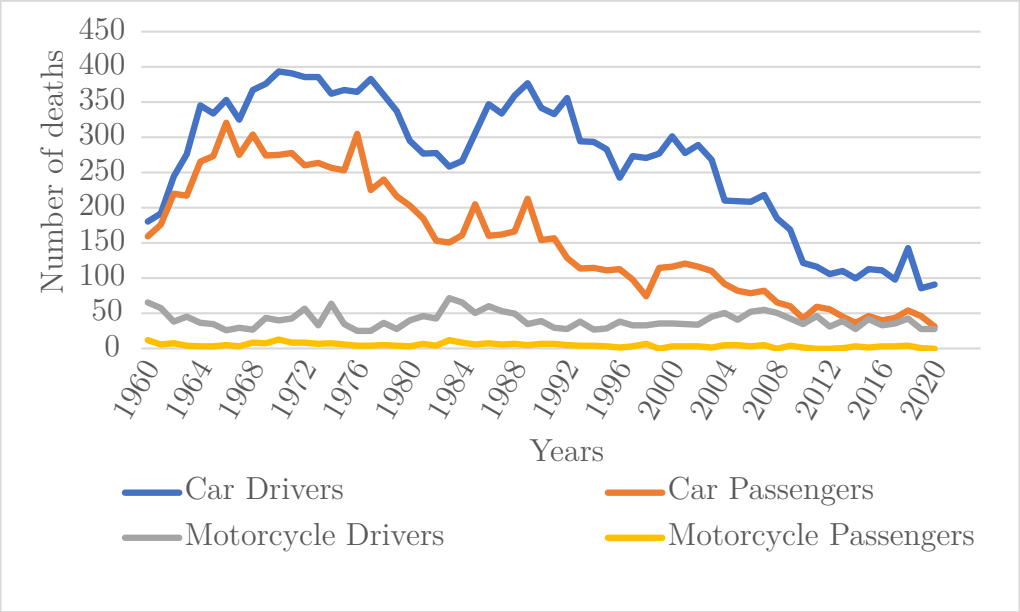


Figure 1.1: The number of deaths of car and motorcycle in Sweden from 1960 to 2020. Drawn based on the statistical data from [1].

in passive protective equipment to trigger such as airbags, and from another side is indeed needed to trigger rescuing alarm for timely medical treatment to improve patient outcomes, especially in the sparsely populated places where the injured driver may not be recognized and given proper first aid, in response to the fact of the high rate of MCC single accidents when driving alone.

There are several products similar to eCall, but some require additional equipment and constant charging which may be inconvenient, see section 2.1 for a detailed literature review. On the contrary, almost every motorist carries a smartphone. Detecht Technologies AB proposed and deployed a smartphone application to detect a motorcycle crash and activate the emergency service to save the life. The Detecht App measures the driving characteristics and sends an alert if a crash is detected. The driving parameters, such as acceleration and rotation, can be used to distinguish severe crash from normal driving, and the data is collected from smartphone's built-in accelerometer and gyroscope. It provides the potential to make motorcycle driving safer and more engaging by adopting an intelligent alerting system. Therefore, it is necessary to improve the performance of the MCC detection solutions.

1.2 Purpose

This thesis is the expansion of two previous theses [7, 10] and other work done by Detecht. They show that machine learning is feasible in motorcycle crash detection using sensors from smartphones. By building on their previous efforts, this thesis project attempts to uncover an automatic method that can detect the crash events as they occur nearly in real-time, and is more accurate than the previous deep learning method.

1.3 Aim

The aim of the thesis is to:

1. develop a machine-learning-based algorithm predicting whether a motorcycle crash has happened in an early phase (before the driver hits the ground).
2. optimize the algorithm to reduce the false positive alarm rate on normal driving and the false negative detection on MCC.

1.4 Scope and Limitations

Evaluation of the algorithm is conducted in all records that have been confirmed as crash events, but only in a small fraction of the normal driving data because training and evaluation on the large number of available driving data require a long computation time. It is tested whether a single simple detector can achieve the desired results. All the confirmed crash records are analyzed, including the true crash events and the false alarms.

Due to the limited time and resources, not all the interesting algorithms are implemented and verified. Similarly, the hyperparameters are manually tuned but the grid search across every combination of possibilities is not involved. Analysis on more simulation or more real crash data would be beneficial for the evaluation of the algorithm.

1.5 Outline

The thesis is organized as follows. Chapter 1 introduces the background and aim of this thesis. In chapter 2 literature review on different anomaly detection methods is summarised and the essential theory to understand this work is provided. Next, the details of implementation and experiments are presented in Chapter 3 as well as the dataset. Then Chapter 4 shows the results and evaluation of the experiments whereas Chapter 5 focuses on the discussion about the domain knowledge, the experiments and results of the algorithms. Finally, the conclusion of this thesis and insights for further work are outlined in Chapter 6.

2

Theory

2.1 Motorcycle Crash Detection

Related researches and patents mainly focus on event-based MCC detection methods, including threshold-based and statistical-based [14]. It means these methods are based on the analysis of MCC dynamics. Threshold-based methods are intuitive. They define a range of sensor values as normal driving, and a large deviation from normal driving ranges is regarded as a crash event. For example, [15] proposed an on-board event analysis that can recognize two scenarios: the motorcycle ending falling (including sliding on the road surface for seconds), and the motorcycle getting stuck vertical standing. This approach can estimate the mounting pose and detect the most likely crash start and end by a mounted e-Boxes with 400 Hz inertial measurement unit (IMU) which detects acceleration and roll angle, and 10 Hz global navigation satellite system (GNSS) which detects speed. Based on the above work, [16] further detects crashes from low to harsh combining off-board severity analysis. The precise event-based MCC detection algorithms heavily rely on sensor calibration because the mounting orientation of the device will influence the inertial measurements [15], and it is difficult to define a generic description clarifying all the cases of MCC. Besides, they usually ignore the temporal dependency between neighboring sampling points, thus detecting only point anomaly introduced in section 2.2.1. Statistical-based methods assume the normal driving follows a multivariate distribution and depicts the anomalies with a significance level (how likely it is that the current output from the sensors is from a normal driving behavior). One example can be [14] which claimed a one-class detection algorithm (a binary classifier) that is insensitive to the driving style. It transforms the temporal signals of IMU into power cepstrum, a type of frequency domain, and detects anomalous when the incoming cepstrum differentiates greatly from a pre-defined reference distribution. Another example is that [17] combines spectrogram and other characteristics extracted from two accelerometers mounted on the bilateral motorcycle body, and claims to distinguish running over potholes and a steered collision through the main and safing judgments. In conclusion, these works show promising results for MCC detection, but also have some limitations. First, they both require dedicated devices. Besides, they are based on handcrafted features that are carefully designed by experts under certain domain assumptions. Finally, they focus on detecting falls or crashes where the vehicles remain vertical standing.

Fewer researches are based on machine-learning methods. [7] trains the one-class

support vector machine (OCSVM) with additional rules on the normal data and tests the algorithm on 5 types of simulated crashes. The results are satisfactory except that it cost a long time to confirm a crash. [18] collects simulated data in three roads: smooth road without road irregularities and crash hazards, irregular road simulating the real city roads, and the road with irregularities and hazards where hitting the hazards is considered to cause a falling crash. Then a two-phased self-organizing map (a kind of neural networks) is trained on the collected data and achieves high Area under the ROC Curve (AUC) under receiver operating characteristic (ROC). This work highlights the importance of distinguishing crashes from crash-like anomalies. There are also some works attempting to detect the crash from the perspective of the classification task. For example, [19] used a supervised Bayesian maximum a posteriori (MAP) classifier to classify the simulated crash dummy tests. Another example is a bicycle crash detection [20]. It has similar signals as that of MCC detection. It manually designs 24 time-domain features and then reduces the feature dimension to 6. The input to the classical SVM is the pre-calibrated and pre-processed data of a time window of 2 seconds. The results achieve 95.2% accuracy but it is only based on the cycling and falling statuses. [21] combines several machine-learning algorithms to classify the normal bicycle drivings and crashes (both simulated and real crashes) with 96.8% sensitivity and 99.6% specificity. The problems on these work lie in the collection of supervised training data that contain mainly simulated dummy tests. It is questionable how the trained classifiers perform on the real-world dataset where noise and other untested cases are often encountered.

Therefore, an MCC detection method minimizing the misclassifications with the high true positive rate (TPR) and low false positive rate (FPR) is crucial, as well as being applicable to the real-world dataset, in order to provide a valuable emergency service.

2.2 Anomaly Detection

Anomaly detection, also known as outlier detection or novelty detection, is defined by Hawkins to find the instances out of the data that "*deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism*" [22]. In this thesis, only the term anomaly is used, since novelties are often identified as unobserved patterns in the normal data, and outliers can be a joint set of the other two. What distinguishes the problem of anomaly detection and classification is the class imbalance of positive and negative samples: it is easy to collect the normal data but anomalies are often rare. The lack of labeled training samples usually leads the anomaly detection to semi-supervised fashion: the model is trained on data without anomalies, and finds out the unusual events dissimilar to the training samples. Anomaly detection can be applied in broad fields [2], such as financial surveillance, industrial fault detection, medical risk in healthcare, and in our case, MCC detection. Normal driving behaviors are common but the rare anomalous behaviors can indicate a motorcycle crash where the driver may need instant rescue.

Deep learning in recent years has outperformed conventional methods significantly in a wide range of applications [2, 22, 23]. The main advantage of deep anomaly detection is the automatic learning of representation features or anomaly scores from the raw data without manual feature extraction by human experts with domain knowledge. This thesis focuses on deep anomaly detection via neural networks.

Due to its rare nature, anomaly detection presents intrinsic complexity compared with classification tasks. Some challenges of anomaly detection have been well solved, while other problems have to be addressed and tackled to achieve the better performance of anomaly detection [2, 22]. Challenge (CH) 1 is the low recall rate for anomaly detection. It is difficult to identify all the rare and heterogeneous anomalies and avoid high false positives on real data. In addition, the distinguishment between normality and anomaly may be blurred in certain contexts, and thus a precise boundary may be hard to define. CH2 is the anomaly detection in high-dimensional and/or not-independent data. In the MCC detection case, the anomalies need to be identified from the instances, on one side, that are temporal dependent within each sensor (accelerometer and gyroscope), on the other side, that are interdependent between sensors. CH3 is the data-efficient learning of normality/abnormality. Since accessible labeled anomalous data is often expensive and hard to collect, semi-supervised and unsupervised methods are more often in practice to learn the expressive representations. Besides, there may also be situations where the labels may be incorrect or inexact. CH4 is the noise-resilient anomaly detection because the real-world data is often contaminated with noise, and vulnerable models may regard these unexpected noise as anomalies. CH5 is the detection of complex anomalies. Most existing methods focus on classifying a single observation whether it is normal or anomalous. However, in our case, it is hard to determine an MCC from a single motion sensor or a single point from the sensor values but requires the joint confirmation from multiple heterogeneous data motion sensors and the consecutive description of the crash dynamics. CH6 is the explainability of the model. It is yet challenging to balance the interpretability and effectiveness of the deep learning models. The current deep anomaly approaches aiming to solve these challenges are summarized as Fig 2.1, and what they are learning are mainly feature representations or anomaly scores.

Among all the approaches for anomaly detection [2, 3, 22], the outputs of these detection techniques are either anomaly scores or binary labels. Binary labels sometimes can be calculated from anomaly scores by setting a threshold. If the output is the representational features through a neural-network-enabled mapping function, then the anomaly scores or binary labels will be computed from the new representation space.

2.2.1 Anomaly Detection for Time Series

Anomalies in time series can be classified into four broad types [3, 22]:

- Point Anomalies: a point anomaly is a specific point that is considered different from the entire time series (global outlier) or the neighboring points (local

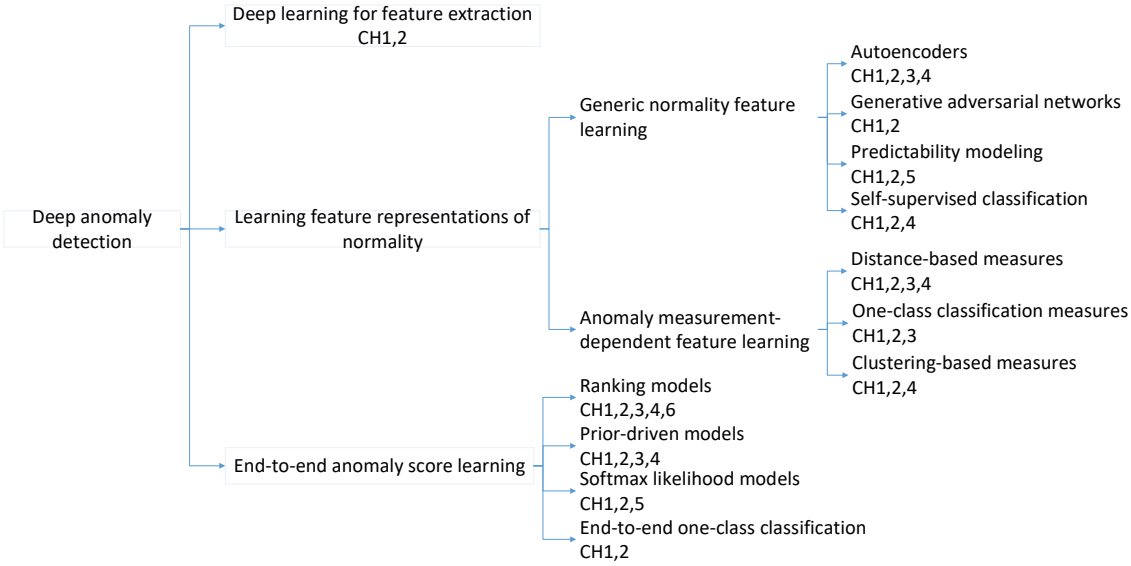


Figure 2.1: Summary of current deep anomaly detection approaches and their aimed challenges. Adapted from [2].

outlier). It can be univariate if it is different in only one variable or multivariate if more time-dependent variables are affected. Point anomalies often happen randomly and particular interpretation may not be possible to make.

- Collective anomalies: also called group anomalies. A collection of individual data points which may not be point anomalies in isolation but exhibit unusual behaviors when joined together is referred as the collective anomalies. In time series it is also known as subsequence anomalies and can be univariate or multivariate.
- Contextual Anomalies: if a data instance behaves anomalously in a particular context but normally otherwise, it is defined as a contextual anomaly, also known as a conditional anomaly. Both behavioral and contextual attributes need to be considered in this anomaly type. For example, if speed provides the context, then large values from accelerometers and gyroscope in a smartphone may indicate a crash if the speed shows that it is under driving, and it may also be the case when a man is shaking the smartphone or mounting it to the handlebar with speed being near zero.
- Time Series Anomalies: if the input data is multivariate time series and a time-dependent variable shows significantly different behavior compared with other component variables, then the whole particular time series is regarded as anomalies. For example, when a sensor is not functioning at one particular axis and this measurement does not reflect the truth but only randomly generated noise.

Anomaly detection for temporal data preserves its unique characteristics, summarized in Fig 2.2. The meaningful type in temporal anomalies is often contextual anomalies, with additional attributes contributing the contexts. Depending on the domain and usage, time is often an anomalous condition, in addition to the above-

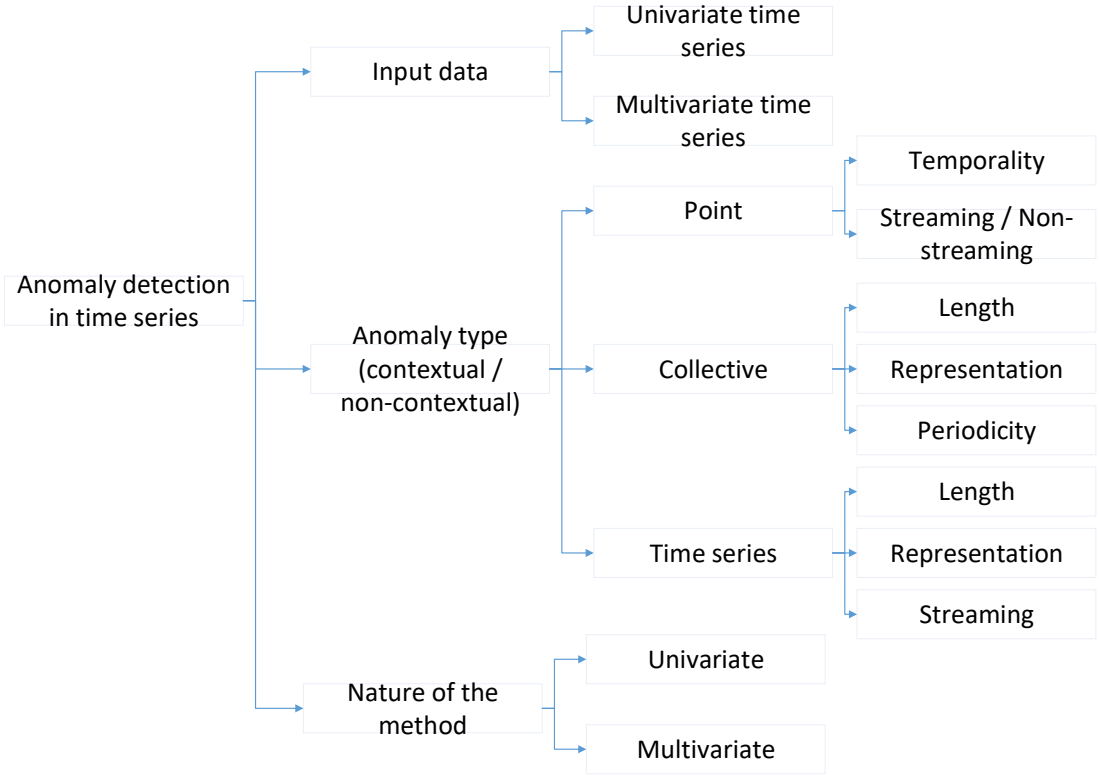


Figure 2.2: Summary of anomaly detection characteristics in time series data. Adapted from [3].

mentioned speed attribute in MCC detection. Another important consideration of temporal contextual anomalies is owing to their dimension and dependency. Temporal dependency is widely addressed in time series, both within a univariate (correlation within one variable) and across multivariate (correlation among variables) time series, to address CH2 and CH4 in section 2.2.

2.3 Neural Networks

A neural network (NN), short for artificial neural network, is a specific machine learning vaguely inspired by the neural networks in animal brains. A more modern term "deep learning" concentrates more on the principles of the combination of multi-scale features, and contains more machine learning frameworks not inspired by neuroscience [24]. The feedforward NN is a classical deep learning model which aims to approximate a certain mapping function $y = f^*(x;\theta)$ where x, y are the input/output to the NN, and θ is a set of parameters that will be learned during the training of the NN. The feedforward NN is trained by backpropagation, which updates the parameters according to the difference between the expected output and the NN output. The word feedforward means there is no cycle connection among its nodes.

Real-world problems are often non-linear functions. It is demonstrated by universal approximation theorem that, a feedforward NN with at least one kind of "squeeze" activation function can represent every mapping function if there are enough number of hidden units within just one hidden layer [24]. However, it is found that this kind of single hidden layer NN may need an unrealistically large number of hidden units. For example, in most cases, the "enough" number of hidden units is the exponential order of n, where n means the n-dimensional input space. Besides, it is found that wide, shallow NNs are often good at memorization, but bad at generalization. To reduce the complexity of NN and make NN practical, different units are designed, together with the ways they connect each other. The units are often called layers in modern NN, and the number of layers constitutes the concept "depth" which refers to deep learning. It is found that deep NNs are better at generalization because it can learn various levels of abstracted features through each hidden layer. In conclusion, deep NNs are demonstrated to have better performance in many application fields. However, the exact depth for each model structure needs to be fine-tuned.

2.3.1 Autoencoder

The autoencoder (AE) is a kind of NN that attempts to get the output as similar as the input. The classical AE is composed of an encoder $h = f(x)$ and a decoder $y = g(h)$, where h is called hidden space, code, latent space, compressed space, representational space, etc. In this thesis, these terms are used to express slightly different meanings. In practice often undercomplete AE is considered, which means the dimension of latent space h is restricted to be smaller than that of input x. In this case, hidden space h is an effective representation of the input and the compressed space is the learned principle component subspace of the training dataset [24, 25]. The learning process of AE can be described as minimizing a loss function $L(x, g(f(x)))$, where L is to penalize the gap between x and $g(f(x))$.

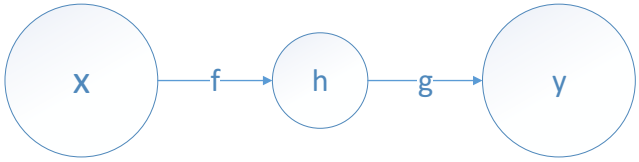


Figure 2.3: An illustration of the autoencoder.

2.3.1.1 Related Work

From a practical perspective, the AE is a popular method with its relatively low cost of training and implementation and straightforward intuitions. The assumption of AE-based methods is that, if the latent dimension is lower than that of input, then the latent space is the low-dimensional feature representations of the training data (normal instances), and normal instances can be well reconstructed while anomalous

instances cannot [3, 22, 25].

Since temporal anomaly detection focuses more on contextual anomalies, the network architecture is correspondingly adapting to the input type and specific domain, such as convolutional autoencoder (CNN-AE), RNN autoencoder (RNN-AE), and combination or variant of them, etc [2, 25]. Based on the assumption of AE, one straightforward way to get the anomaly detection results is to regard the reconstruction error (sometimes called prediction error in RNN domain) as anomaly score, and then use the deviated value from a fixed threshold as the confidence score or to classify the instance as normality or anomaly. The fixed threshold is a pre-defined value after training. However, there is no such method that dedicates to dynamic and adaptive thresholds in univariate/multivariate in collective outliers [3]. The other way is to make use of the learned representations of AE. For example, after training a CNN-AE, one can extract the outputs of only the encoder and let the one-class classifier tell if the hidden representations are normal or anomalous. The one-class classifier can be the one-class support vector machine (OCSVM), support vector data description (SVDD), or more advanced methods such as the end-to-end one-class classifier generative adversarial network (GAN) [3]. Another example would be that one may also use RNNs to build a prediction model and the prediction error would be the difference between learned hidden representations and predicted representations [26].

One significant advantage of AE-based anomaly detection is that there are a variety of powerful neural network variants available (CH2), and then it may learn more expressive features than those handcrafted by human experts to help reduce the false positive rate (CH1) [2]. On the contrary, CH5 is a big issue in AE-based techniques because on one hand, the expected infrequent regularities may not be learned and the unexpected presence of outliers in the training data may be learned [2]. On the other hand, the loss function aims at learning the underlying regularities, rather than directly learning the anomalies [2].

The window-based AE is a widely used variant of AE in accelerometer data analysis [2]. The input samples are overlapping sliding windows of multivariate time series, illustrated in Fig 2.4. Through AE, the temporal dependencies are modeled within the input time window, so this method is only suited for relatively simple time series, but not for data that have long-range temporal relationships [27].

CNN is a popular structure to construct the representational space, and there are also several other domain-specific structures other than CNN, such as recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), etc., that are popular in complex time series. For example, [27] found CNN-AE performed comparable with long short-term memory autoencoder (LSTM-AE) on machine temperature and ECG datasets, and [28] found similar results on NAB and S5 datasets, but in certain dataset, the latter one outperforms the former. It also applied to [29] where there is no distinction between CNN-AE and LSTM-AE in certain datasets. [30] summarized that CNN-VAE may be suitable for deployment

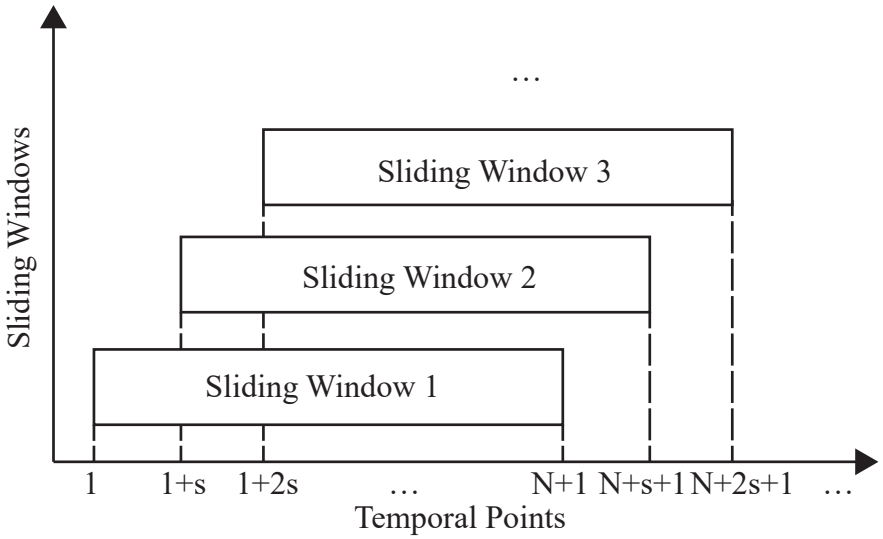


Figure 2.4: Creating sliding window samples for anomaly detection. The x label is the temporal sampling points, the time window is created in an overlapping style, where s is the stride, N is the length of the sliding window.

in edge devices. [31] used DNN-LSTM as predictability modeling and the deviation values as anomaly scores on a run-to-failure bearing dataset. [32] changed multi-channel CNN to multi-head CNN and then concatenated RNN to obtain deviation scores. Besides, in addition to AE, there are GANs, and other learning methods available and rapidly emerging, as summarized in Fig 2.1. It is unknown which model architecture is optimal for the current case before trying every model, but it is suggested to test the simpler models first unless they do not yield desired performance [27, 33]. Previous work by Detecht [7, 10] show that detecting collective anomalies based on raw data space can be troubled by the high false positive rate but neural networks can be promising, so it is reasonable to investigate more on contextual anomalies via neural networks based on previous work.

2.3.2 Convolutional Neural Network

CNN is the first kind of NNs that solves important business applications [24]. The word "convolutional" means the operation "convolution" is applied to at least one layer of this NN, instead of conventional matrix multiplication. The convolution operation in CNN is actually cross-correlation, which is not commutative compared with conventional convolution operation in the function of real variables. Convolution is calculated as equation 2.1, where S is the output, I is the input, and K is the kernel. In practice, K is much smaller than the input I to achieve sparse interaction [24] and produce the compressed feature map.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.1}$$

A basic convolutional architecture is usually composed of three building blocks, as shown in Fig 2.5, a convolutional layer to apply affine transformation, a non-

linear activation function to introduce non-linearity, and a pooling layer producing the unit results by the overall statistical features of neighboring output at certain locations. The pooling layer is like a prior assumption, that the learned function can be invariant to slight shifts.

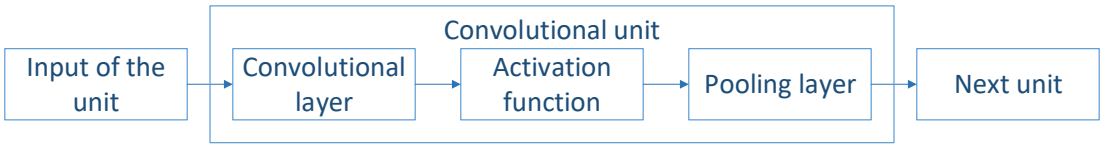


Figure 2.5: A basic convolutional unit

The above unit is applied to the encoder. The decoder has a similar structure as the encoder, but the building blocks are slightly different. The deconvolutional unit consists of a transposed convolutional layer (also called deconvolution or upsampled convolution), a non-linear activation function, and an up-sampling layer, illustrated in Fig 2.6. On the contrary of convolution that compresses the input values, the transposed convolution outputs a larger shape through the kernel by broadcasting.

Figure 2.6: A basic deconvolutional unit

3
Method

3.1 Anomaly Detection Method

In this thesis, training the anomaly detection algorithm consists of two steps. In the first step, a CNN-AE is trained on normal dataset to learn the compressed features of normal driving and to reconstruct the normal driving behaviors. Then, the second step reconstructs all the data samples, including normal and crash datasets, to produce the anomaly scores derived from the gap between the reconstruction error and a tuned threshold, details in section 3.1.3.

3.1.1 Assumptions

The principle assumption inside the AE method is that it can learn the normal driving behaviors if trained on the normal driving data so that the reconstruction error of normal samples is small. On the contrary, the reconstruction error of the crash samples will be large since the model did not learn the crash dynamics. Hence a properly tuned threshold of reconstruction error can differentiate the anomaly from normality as well as possible. The tuning of the threshold is introduced in section 3.1.3.

3.1.2 Algorithm Scheme

The proposed crash detection scheme has two phases. Only passing both phases, will a crash alarm be triggered. Phase one is a simple event-based method. The aim of phase one is to reduce the false positive rate (FPR), by filtering out the cases when the smartphone is moving very slowly or has small accelerations or rotations. It is expected that the FPR can be reduced by making full use of available information. It works by comparing the measured sensor values with pre-defined thresholds. If the sensor values exceed the thresholds for a while, then it indicates that the driver is in a considerable movement. If the sensor values are below the thresholds, then phase two will not be triggered, and the whole scheme will not raise a crash alarm. If phase two is triggered, then the crash detector (a CNN-AE) will check if it is a crash event by comparing the reconstruction error with a pre-defined threshold (TH_A) and the number of consecutive anomalous samples with a pre-defined threshold (TH_B).

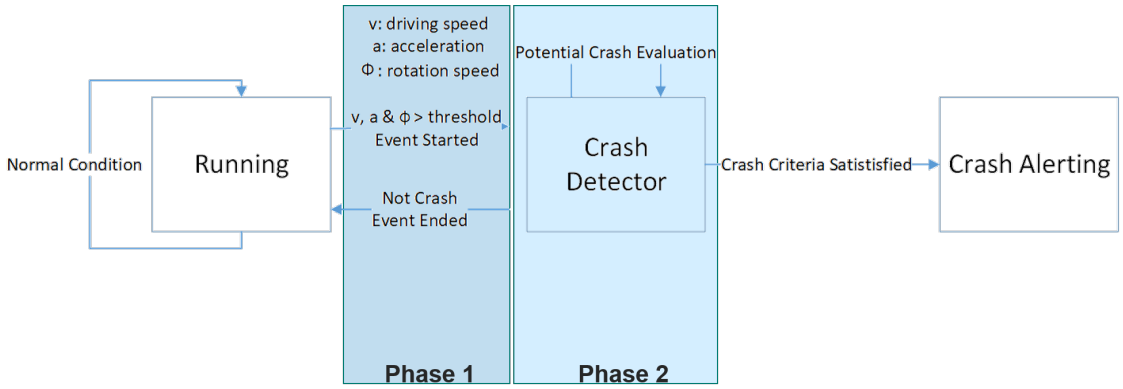


Figure 3.1: Block diagram of the state machine of proposed MCC detector. a and ϕ is the measured acceleration and angular velocity, respectively. PSAP, the Public Safety Answering Point, is the emergency service provider in Sweden.

3.1.3 Model Tuning

After the reconstruction model is trained, the algorithm of anomaly detection requires another five additional parameters: the thresholds set for driving speed, acceleration, rotation speed in phase one, and the threshold of reconstruction error TH_A to define a point anomaly and the number of consecutive anomalies TH_B to define a collective anomaly in phase two. These five thresholds are obtained by balancing between the performance of anomaly detection and the frequency of false positive alarms, which will be explained in the following paragraph. For the parameters in phase one, first, the false positive events are statistically analyzed when the phase one parameters are set to zero. Then, these three parameters are set to the Euclidean norm of the smallest sensor values among these false positive events.

As for the parameters in phase two, if TH_A is simply set as the maximal error in the normal training set, then it would be sensitive to the noise and the most extreme and suspicious driving behaviors in the training set but still did not detect a potential crash. In practice, it is common to use the 99% percentile (the location in a distribution) or the 75% percentile + 1.5 $*$ interquartile range (the difference between 75% and 25% percentiles) of the reconstruction error in the training set to eliminate the influence of outliers [34]. The algorithm will become a point anomaly detector if TH_B is set as 1. Therefore, a grid search method combining the tuning of TH_A and TH_B is adopted in this thesis. First, a range of $91-100\%$ percentile of reconstruction error is compared together with the TH_B ranging from $50-200$ to find out a coarse performance. Then, a finer classifier is tuned with TH_A ranging within $98-99\%$ and TH_B ranging within $1-250$ to get their comprehensive profiles by AUC. Finally, a pair value of TH_A and TH_B is decided for each model according to the results. As for the triggering thresholds for the three contextual attributes, they are tuned and searched from the parameter candidates lists by analyzing the difference of the sensor values in the anomalous event and true labeled crashes. The potential issue of this grid search method lies in that during the training phase, the test set (crash data) are usually separated from the training and tuning phase,

which aims at reducing the effects of overfitting [24]. However, it is not a practical approach in this thesis because the number of the crash set is so few that dividing the crash set is not ideal. From another point of view, it is a common practice to utilize all the available data into the model training once the approach is known to be valid. Therefore, it should not be a serious problem. Besides, the sensitivity and specificity of the model should both be as higher as possible when tuning the model. This thesis strives to let the sensitivity as 1 because detecting all the crash events can save lives as much as possible.

To visualize the joint effects by the TH_A and TH_B, the ROC curve is adopted, where the definition of the false positive rate in Fig 4.2 is slightly different from equation 3.7. If the FPR is conventionally calculated based on the number of misclassified samples, the whole curve would squeeze at a small corner in the lower right because the sample FPR is below 1% in the normal logfiles. For a better effect of visual illustration, the FPR here is calculated by equation 3.1.

$$FPR_{ROC} = \frac{the\ number\ of\ logfiles\ raised\ false\ alarms\ by\ the\ thesis\ algorithm}{total\ logfiles\ examined} \tag{3.1}$$

3.2 Data

3.2.1 Dataset

The dataset is provided by Detecht Technologies AB. The data is collected by the smartphones from Detecht App users as logfiles. There are users pretty much all over the world, such as the US, Europe, and Australia, etc. The majority of data is during normal driving, and a rare number of logfiles is confirmed to contain only one crash event, but the exact crash time when an MCC happens is unavailable. The smartphone is recommended to be mounted as Fig 3.2. It is also acceptable to have it in a tight pocket as long as it is stable and does not bounce around (like in a loose bag). However, the data is not guaranteed to be collected in these configurations.

3.2.2 Features

Detecht App, available on both iOS and Android, uses the built-in sensors in smartphones to collect data. The features used in this thesis include sensor values from accelerometer, gyroscope, and estimated speed derived from GPS signals. There are each 3 variables, X, Y, and Z directions in the accelerometer and gyroscope sensors.

Accelerometer and gyroscope sensors are expected to be robust and disturbance-free measurements regardless of the smartphone models [7]. The information from these six variables is used to predict whether an MCC happens. In addition, speed is another criterion to confirm an MCC because on the one side, it is meaningless to trigger a crash alarm when the user is under low-speed state, such as standing still shaking the smartphone, walking, or mounting/dismounting the smartphone. On

Figure 3.2: Recommended configuration of the smartphone mounting. Source: Detecht.

the other side, acceleration after slow-down within seconds may indicate the driver hit a bump or a pothole but without crash happening [15].

3.2.3 Sampling Rate

The sampling rates are varying among different features and different smartphone models. Irregular sampling rate might be a problem for the crash detection algorithm. Exploratory data analysis shows that for measured values from accelerometer and gyroscope, the majority of the sampling rates is 100 Hz as expected [35], and the time interval distribution is plotted in Fig 3.3(a). The sampling intervals within 5-15 ms account for 97.27%. Few sampling intervals are extremely large and unrealistic (e.g., 1592447824476 ms) but their occurrence is rare (5 intervals in the total crash set) so these irregular sampling intervals are manually changed to 10 ms. It is guessed that these large sampling intervals are due to the missing timestamps when sampling. Besides, there are rare negative sampling intervals, and they are also manually set to 10 ms. For the estimated speed from GPS, the majority of sampling rate is 1 Hz, where Fig 3.3(b) plots its distribution. The sampling intervals within 500-1500 ms account for 90.59%. The irregularities may come from 4 aspects. On one hand, the capacity of processing unit in different smartphones can vary thus not all phones can keep the sampling frequency as designed. On the other hand, the resource allocator of a smartphone may not respond to the Detecht's App on time since there might be several mobile applications running simultaneously. In addition, GPS signals may not always be accessible, which may contribute to the greater irregularities on the speed sampling intervals. Finally, it is found that smartphones are more likely to sample a longer time interval on lower speeds.

However, there is no need to especially tackle this issue for several reasons. Firstly,

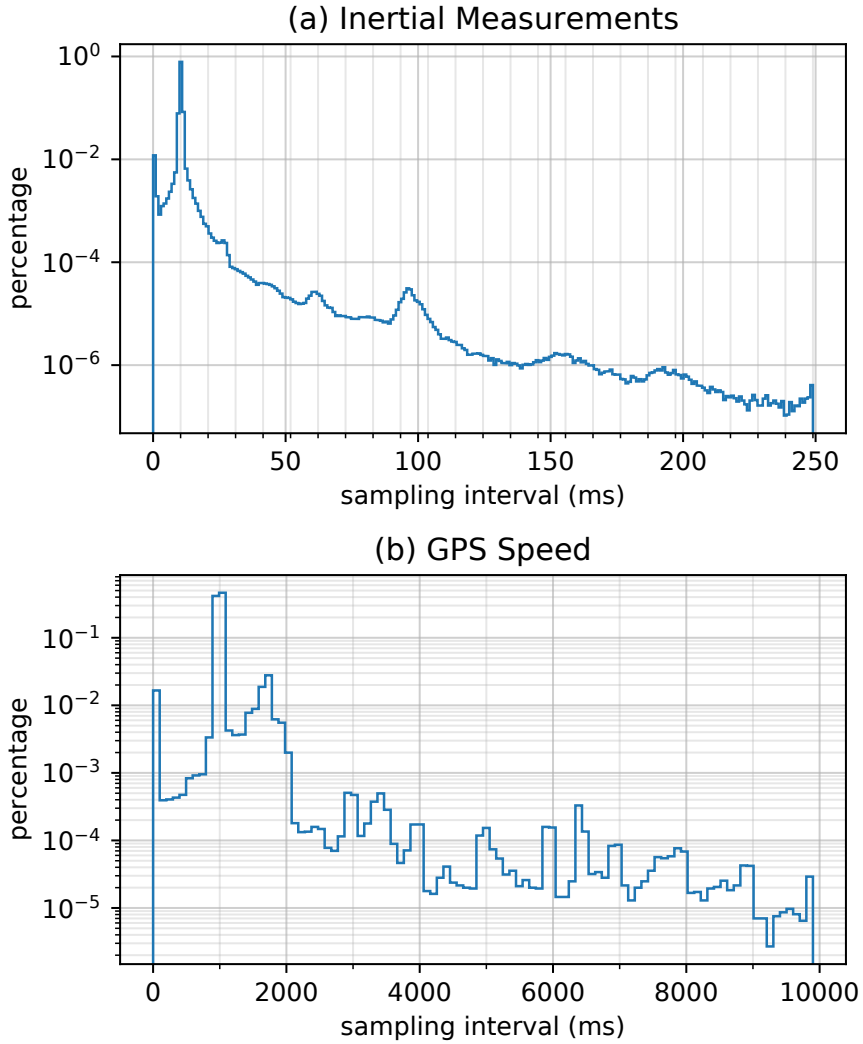


Figure 3.3: Distribution of sampling intervals of different sensors.

97.27% of the sampling rate from the inertial sensors is focused around 100 Hz. According to the literature [36, 37, 38, 39], irregularly sampled intervals only raise attention when the drop rate (the total frequency of irregular intervals) is between 10% to 90%, while the drop rate of Detecht's dataset is 2.73%. In addition, since irregular sampling intervals can be observed in practice, taking these irregularities into consideration may help increase the robustness of the MCC detection algorithms.

There are some duplicated lines where the sampling interval is 0 ms, which may be due to writing buffering to the CSV logfile on smartphones, according to Detecht's analysis. Its effects can be ignored in the model training for the two following reasons: it only accounts for about 1% of the total sampling intervals, and in correct recordings it is common to see two neighboring lines are identical, even with valid timestamps.

3.2.4 Data Exclusion

According to the instructions of Detecht App [40], the smartphone should be mounted on the handlebar or worn tightly on the body of the driver while riding. However, it is unknown whether the driver followed the instructions thus the measured data may not always be reliable. There are five crash logfiles excluded from the testset where one logfile lacks the recording header and the rest four logfile are duplicated logfiles because these four have the complete same contents as other four logfiles. In summary, there are a total of 500 normal logfiles and 17 crash logfiles.

3.2.5 Algorithm Performance

The estimated timestamps of the crash events in the crash set are provided by Detecht's confidential algorithm. The crash events detected by the thesis algorithm were checked with the crash timestamps provided by Detecht. For both normal driving logfiles and crash logfiles, the FPR (equation 3.2, is another form of specificity where $specificity = 1 - FPR$) is emphasized for the convenience of drivers. To evaluate the algorithm's capacity for detecting true crashes, only the number of estimated true crashes are concerned because the total number of true crashes is fixed once the crash dataset is defined. The evaluation metrics are summarized in section 3.5.

$$FPR = \frac{FP}{TN + FP} \tag{3.2}$$

3.2.6 Data Standardization

There are several ways to normalize or standardize the data. To encourage different variables to share the same weight when training the neural network, Z-score standardization is adopted in this thesis to give these variable a same mean of 0 and variance 1, as equation 3.3. The mean and variance are computed from the training set as the sample mean and sample variance, and applied to the entire dataset, including training set, validation set, and test set, as a common practice.

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

3.2.7 Fixed Size Time Window

The MCC detection algorithm should be able to detect the contextual anomalies in time series while neglecting the point anomalies. In other words, the algorithm should be able to capture the temporal dependencies of normal driving and not being sensitive to sudden changes within a very short time (e.g. 10 ms as a sample) which may be caused by road anomalies, or fluctuations of sensor measurements. In addition, an MCC often lasts for milliseconds, so the window-based method is suitable for this kind of tasks where only several recent inputs contain relevant information. Besides, in order to reduce the false alarms for MCC, it is necessary to confirm the event of a real crash by analyzing a longer period of sensor information.

Therefore, the time-window-based method is adopted. A sliding time window of certain samples after each record is created as a new sliding window sample and fed into the MCC detection algorithm as the input data. The number of certain overlapping samples is a hyperparameter and determines the length of temporal dependencies the MCC detection algorithm can learn [27]. The length of the time window is firstly set as 30 sampling points [10] to constitute a sliding window sample of mainly 300 ms. Another length 128, corresponding to roughly 1280 ms, are tested to find the optimal setting. Then the joint-optimization with the number of consecutive collective anomalies is conducted. It would be beneficial to search for the optimal length of the time window, but due to the restriction of the length of this thesis, this search is not performed. It is hypothesized that the crash dynamics might be better captured within a longer comprehensive time window. Besides, it is a common practice to set the batch size of a power of 2 for the reason of accelerating NN computation [41], so 1280 ms might be better.

3.3 Implementation Framework

This thesis is implemented with Python programming language and Keras. Keras is an open-source platform, which is integrated as a high-level module in TensorFlow 2.4.1. Keras was chosen for its fast prototyping and relative simplicity to implement different model structures. To accelerate the training process, multi-processing on GPU was adopted. The training GPU is NVIDIA Quadro P5000.

3.4 Convolutional Autoencoder Architecture Details

3.4.1 Autoencoder

There is no gold standard CNN-AE capable of general applications. In this thesis, the design of encoder and decoder borrowed some classical ideas from CNN architectures and modified them to tailor our data. Encoder is the backbone of AE to compress the input data to a latent space with lower dimensions. In this thesis, two types of backbones are explored: VGG style autoencoder (VGG-AE) and mobileNetV3-small style autoencoder (mobile-AE). mobile-AE produced the main results, and VGG-AE provided the supplementary analysis. The decoder is the reconstruction head of AE. In this thesis, all decoders are vanilla fully convolutional neural networks.

VGG, short for Visual Geometry Group in Oxford University, is one of the simplest but yet powerful CNN structure. It is featured with cascaded convolutional and max-pooling layers to extract a hierarchical feature map. Its unique improvement compared with its ancestor is the 3×3 kernel-sized filters with deeper layers replacing larger and shallow ones to introduce more non-linearity. Since the original VGGNet is designed for image data, the customized VGG-style AE in this thesis in shown in Fig 3.4. MobileNetV3-small is a small version in the mobileNetV3 family with fewer

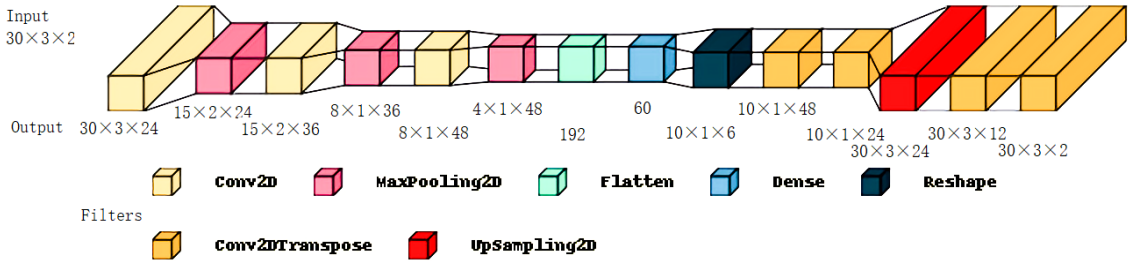


Figure 3.4: VGG style AE customized to MCC crash detection.

MobileBlocks and filters. The MobileBlocks, also named bottleneck layers or bneck for short, are composed of three steps: 1×1 input convolutional channel, 3×3 or 5×5 convolutional channel, and another 1×1 input convolutional channel. The detail of the tailored encoder is listed in Table 3.1 The exp size is the number of channels after inverted residual with linear bottleneck unit, and the *out* is the number of channels to the bneck layer. SE is short for squeeze-and-excitation networks, which squeeze the features with less importance and excites those more important features. NL is short for nonlinearity, the type of non-linear activation function used in this operator. HS is short for hard-swish (equation 3.4) and ReLU6 is similar to ReLU but limits the maximal output to 6 (equation 3.5). The s is the stride for the current block. The hyphen '-' means not applicable.

$$\text{h-swish}\ (x) = x \times \text{ReLU}\,6(x+3)/6 \tag{3.4}$$

$$\text{ReLU6}(x) = \min(\max(0, x), 6) \tag{3.5}$$

Input	Operator	exp size	#out	SE	NL	s
30×3×2	conv2d, 3x3	-	16	-	HS	2
15×2×16	bneck, 3x3	16	16	√	RE	2
8×1×16	bneck, 3x3	72	24	-	RE	2
4×1×24	bneck, 3x3	88	24	-	RE	1
4×1×24	bneck, 5x5	96	40	√	HS	2
2×1×40	bneck, 5x5	240	40	√	HS	1
2×1×40	bneck, 5x5	120	48	√	HS	1
2×1×48	conv2d, 1x1	-	96	√	HS	1
2×1×96	pool, 2x1	-	-	-	-	1
1×1×96	conv2d 1x1, NBN	-	96	-	HS	1

Table 3.1: MobileNetV3-small style encoder customized to MCC crash detection. The decoder structure is the same with VGG-style AE but a different number of hidden features (VGG:60, MobileNetV3-small:96).

3.4.1.1 Loss Function

Loss function is the function that penalizes the decoder output $\hat{x} = g(f(x))$ for being dissimilar from the input x. The dissimilarity is represented by the residual features. There are several types of residual features proposed but in this thesis only

the most widely used mean squared error (MSE) is adopted as the loss function and for accuracy evaluation of AE reconstruction [42].

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2 \tag{3.6}$$

3.5 Evaluation Metrics

There are different evaluation metrics corresponding to each process of training the whole MCC detection algorithm. When predicting the output of the input data, the output of AE is sent to compare with the input to see how well the AE reconstructed the input. MSE is adopted here, the same as loss function, to measure the dissimilarity between raw data and prediction.

To evaluate the performance of the models, the FPR in the normal and crash logfiles is calculated by equation 3.7. The "samples" is created by the sliding window method described in section 3.2.7. The specificity is simply $1-FPR$. The other metric is the TPR calculated by equation 3.8, also known as sensitivity. TPR is only applicable in the crash logfiles since only they contain crash events (corresponding to the true positive alarms). FPR is calculated for the normal and crash logfiles, respectively.

$$FPR = \frac{the\ number\ of\ samples\ that\ raised\ false\ alarms\ by\ the\ thesis\ algorithm}{total\ samples\ examined} \tag{3.7}$$

$$TPR = \frac{TP}{TP + FN} \tag{3.8}$$

4

Results

The presented main results are based on mobile-AE with the time window 300 ms trained on a larger subset of normal logfiles. The supplementary experiments include1 one mobile-AE model and 3 VGG-AE models. The supplementary mobile-AE model is trained on the larger subset of normal logfiles, but with a different time window of 1280 ms. The 3 VGG-AE models are all with a time window of 300 ms, where one VGG-AE model is trained on a larger subset of the normal logfiles, and the other two models are trained on a smaller subset of the normal logfiles, in which one is with different crash detection parameters.

4.1 Main Results: Mobile-AE

4.1.1 Model Tuning

The distribution of reconstruction error among training, validation, and test sets is visualized in Fig 4.1. In this chapter, because most of the reconstruction errors are small, the reconstruction errors higher than 20 are cut out for a better visual illustration effect in graphs. It is clear that all the three subsets follow a positively skewed distribution where the right tail of the distribution is fat. The distributions of three subsets focusing on small reconstruction errors suggest that the model mobile-AE learned the training set and generalized well to the validation set and the test set. Compared with normal driving logfiles, the crash logfiles occupying a heavier portion in the large reconstruction error means that it is hard for the model to reconstruct the unlearned crash patterns, thus making it possible to define a threshold distinguishing the normal driving parts and crash events.

The ROC curves are plotted in Fig 4.2 where the tuning threshold is the TH_B, so the number of consecutive anomaly samples, known as collective anomalies, counts here. From the ROC curves, it looks like that the 99% percentile has the better generalization capacity to have a higher true positive rate and lower false positive rate. However, the 99% percentile condition missed one crash event by analyzing the numerical results, which indicates that the 99% percentile as the threshold for the reconstruction error is too high. In order to detect all the crash events (true positive rate = 1), the 98% percentile was selected as the threshold. Since the false positive rate is relatively high, additional methods have to be utilized to further reduce the false alarms beyond the collective anomalies. It is worth spending time on finer tuning the percentile condition when applying this algorithm into practice, e.g.

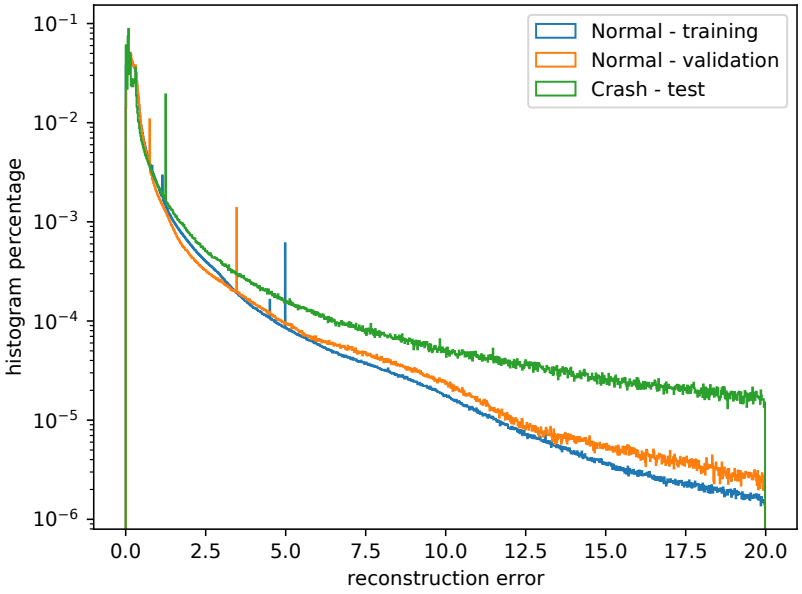


Figure 4.1: Distribution of reconstruction error of mobile-AE trained on the larger subset of normal logfiles.

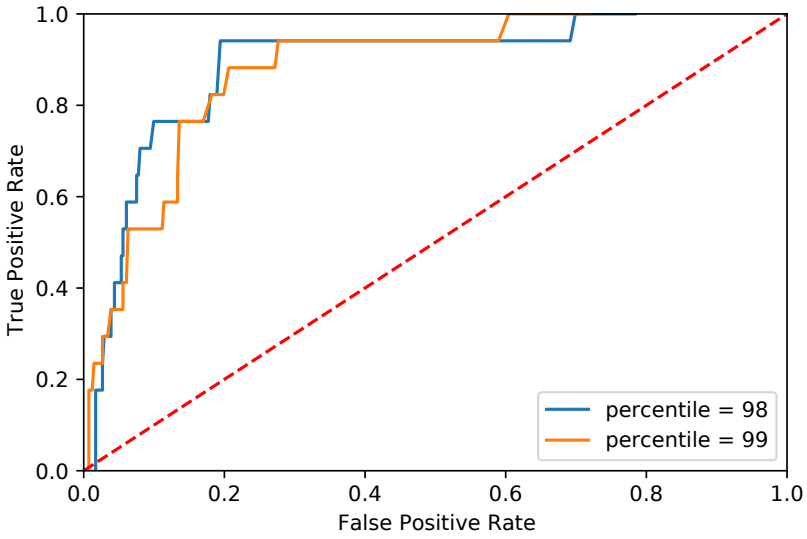


Figure 4.2: The ROC curve of mobile-AE. The percentile of reconstruction error is based on the training set. The false positive rate and true positive rate are calculated from the training set and test set, respectively.

from 98.01% to 98.99%, but it is not performed in this thesis due to the lack of time.

As introduced in section 2.2.1, speed is the contextual attribute to define a crash event, contributed as contextual anomalies. As defined in section 3.1.3, raw sensor values from the accelerometer and gyroscope are used to trigger an anomaly event detection, so there are two additional contextual attributes in the MCC detection. In addition, the speed attribute is also used to cancel a false alarm if the driver is still driving in the next few seconds after an alarm is triggered. These two parameters (definition of "still driving within a few seconds") are only coarsely tuned.

4.1.2 Model Performance In Detecting Crash

The main results are based on a set of tuned parameters. Under this set of parameters, all the crash events are correctly identified by the proposed algorithm, so the TPR (sensitivity, or recall) is 100% in this setting. One example is Fig 4.3, where there is a starting gap between the algorithm proposed in this thesis and Detecht's current algorithm. It is because of that, on the one hand, the thesis algorithm needs the previous 300 ms windowed samples to start an anomaly monitoring event while the Detecht's algorithm runs continuously. On the other hand, the proposed algorithm only detects the crash undergoing time while the plotting of the Detecht's algorithm contains a longer confirmation time for a better illustration. The undergoing time of the proposed algorithm is close enough to the start time defined by Detecht's current algorithm (within 200 ms), as visualized in the zoom-in Fig 4.4.

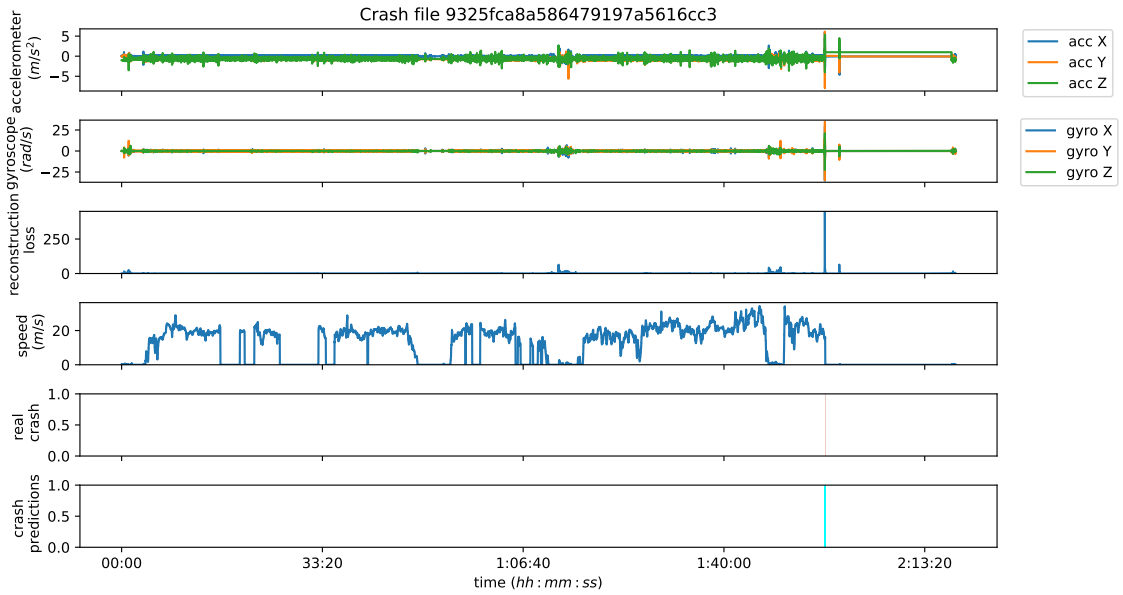


Figure 4.3: Example of crash detection.

In short, in 7 simulated crash logfiles, the proposed algorithm performs quite well, with all crashes detected correctly and no false alarms (sensitivity = 100% and specificity = 100%). In the rest 10 real crash logfiles, the algorithm performs as well

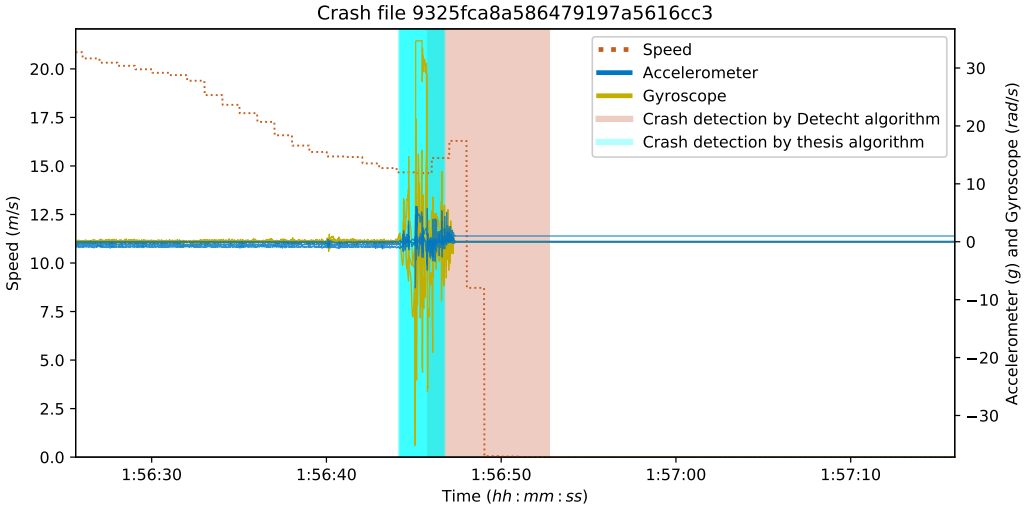


Figure 4.4: Zoom-in example of crash detection.

as in the simulated crash events in 1 file (Fig 4.3) with unit sensitivity but raises a few false alarms in other 9 crashes which will be examined below.

4.1.3 Model Performance In False Alarms

The specificity in normal drivings is based on the training set because a larger subset of normal drivings is more representative for general case, and the results are presented in table 4.1.

Evaluation Metrics	Crash Logfiles	Normal Logfiles
Specificity	98.7715%	99.9896%
False Positive Rate	1.2285%	0.0104%

Table 4.1: Specificity and false positive rate in crash logfiles and normal drivings in the main results of mobile-AE trained on the larger subset of normal logfiles. It corresponds to one false alarm per 1.8 driving hours on average.

In the normal logfiles, there are in total 186,661,622 samples examined, and 19,362 samples are misclassified as a crash event (0.0104%), causing 287 false alarms in 51 driving logfiles. A more intuitive description is that: the proposed algorithm will raise one false alarm per 1.8 driving hours in average. Inspecting the detail of false alarms reveals that some users are significantly more likely to trigger false alarms than others. The vast majority of all users never trigger an alarm, but a small subset often triggers several false alarms with relatively little driving. In detail, 35% (18) of these 51 false alarms logfiles contribute 80% (229) of the all false alarms, who raise more than 3 false alarms. They exhibit quite noisy records, where one example is drawn in Fig 4.8. It might be due to the driving, but it is also possible to be caused by the sensor measurement errors. For example, in the Fig 4.8 when the three false

alarms are raised, their gyro X are comparably much larger than their neighbors. Further examination reveals that 12%(6) logfiles contribute 55%(157) false alarms who raised more than 10 false alarms per logfile. Barring these 6 logfiles will result in one false alarm per 4.0 recording hours while barring these 18 logfiles will result in one false alarm per 8.9 recording hours. However, it is most reasonable to keep these records in this analysis for two reasons. From one side, noisy records can sometimes be in the real-world scenario, and the algorithm performance should count in all the available datasets. From the other side, these noisy records should only be excluded once other methods are taken to reduce the effect of abnormal patterns, such as noise repairing discussed in section 5.2, or a message telling the users that their abnormal recordings will cause many false alarms.

Another interesting result from table 4.1 is that the false positive rate in crash logfiles are much higher than that in normal logfiles. A possible cause of the difference is the size of the samples: the number of normal logfiles calculated here is over 24 times more than that of crash logfiles (412 v.s. 17). Visual inspecting of crash logfiles reveals 8 noisy logfiles, accounting for 47% of all the crash logfiles. Therefore, from the view of statistical power, the specificity of normal logfiles should be emphasized.

One example crash logfile containing false alarms is in Fig 4.5. It can be seen that there are several false alarms in around 40 minutes of recording. Its crash section is zoomed in in Fig 4.6. Another example is a normal logfile, as shown in Fig

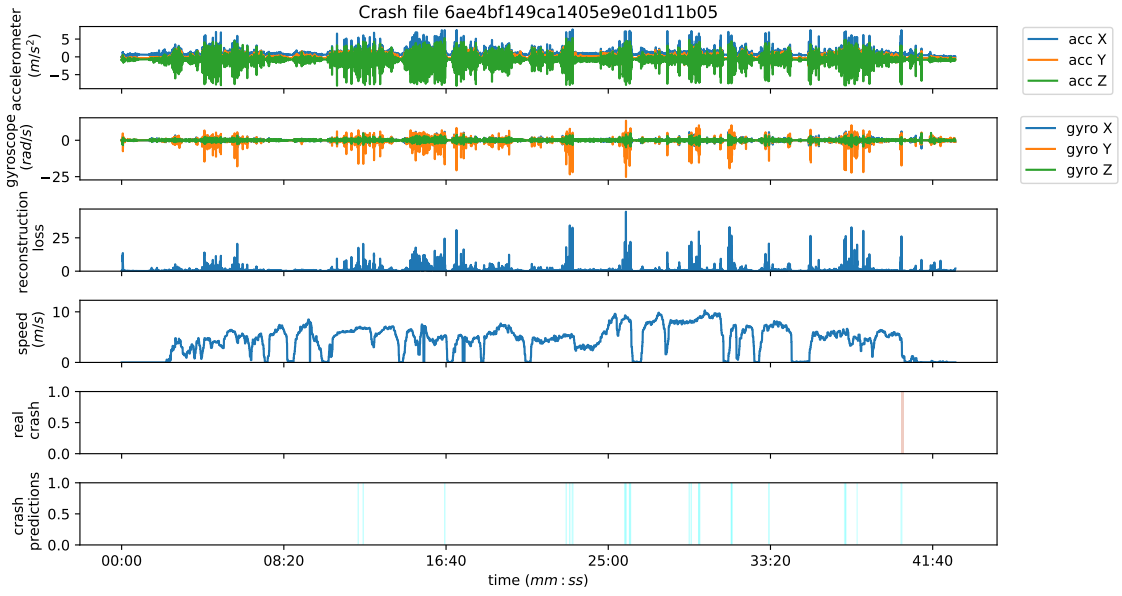


Figure 4.5: Example of false alarms in the crash set.

4.7. It shows a classical pattern of false alarms which can be easily eliminated by monitoring the speed for a longer time period. For example, if so the alarm can be automatically cancelled after waiting for 10 seconds by finding the driver is still driving with low but non-stop speed.

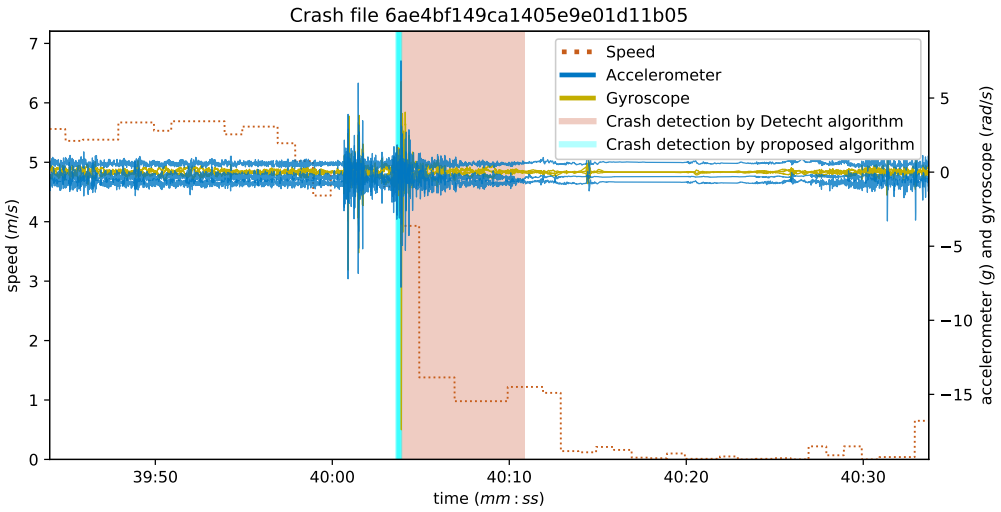


Figure 4.6: Zoom-in example of false alarms in the crash set.

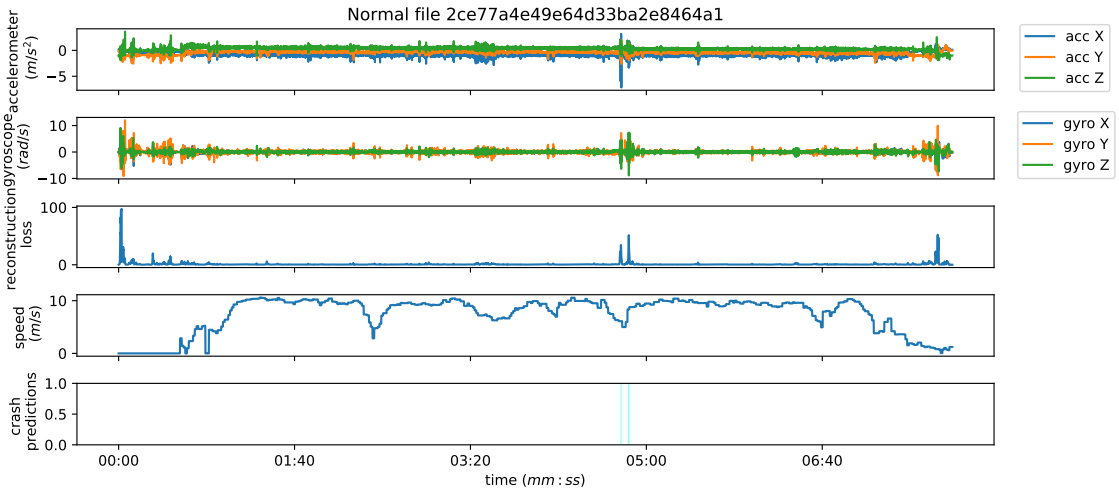


Figure 4.7: Example of false alarms in the normal set.

4.1.4 Inspecting Abnormal Logfiles

To better understand the false alarms, logfiles that produce false alarms are examined individually to see why the MCC detection algorithm performs much worse on them than others. Besides the above-mentioned two false alarms logfiles, a classical pattern that is likely to raise one false alarm is when the sensor values of accelerometer and gyroscope are abnormal. It suggests that the accelerometer and gyroscope sensors are not as robust as we would hope for. For example, in Fig 4.8, there are several samples where the gyroscope values exceed $\pm 10\ m/s^2$. These false alarms are not canceled by the proposed algorithm because the current canceling criteria is relatively strict and simple: the definition of "still driving" is the speed higher than a tuned threshold for the following 1.5 seconds. Improving such definitions such as adapting the thresholds for the individual users might help eliminate such false alarms. Another example is in Fig 4.9, where the acceleration and angular velocity can exceed $\pm 5\ m/s^2$ and $\pm 20\ rad/s$ for several minutes, respectively. The detail of true crash of this record is visualized in Fig 4.10.

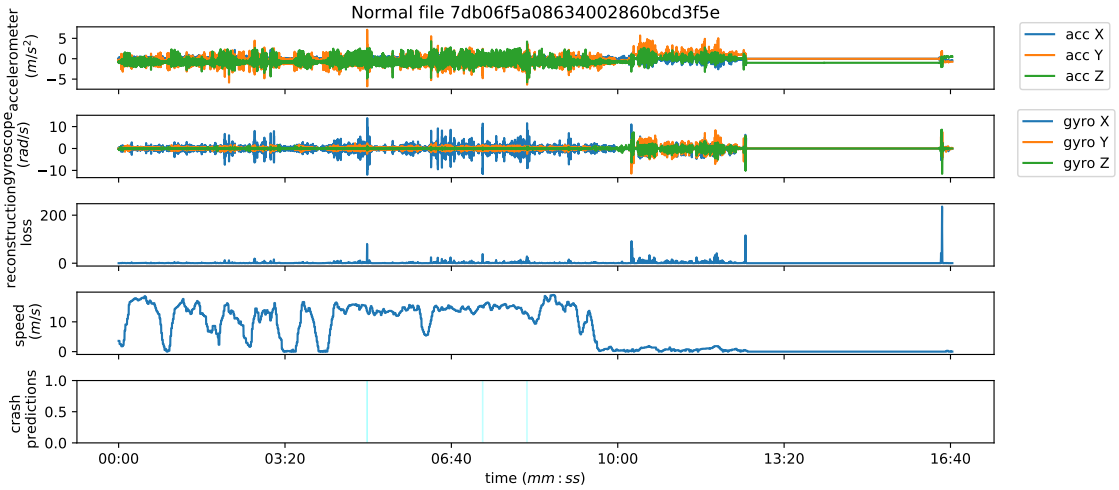


Figure 4.8: Example of the false alarms in the normal set.

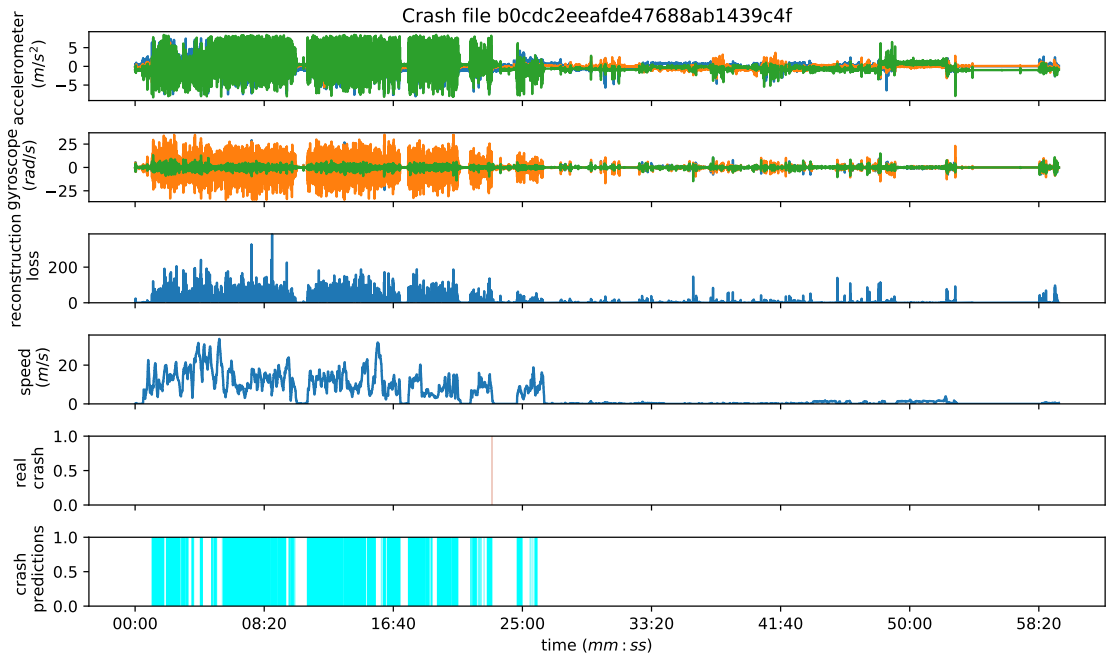


Figure 4.9: Example of a lot of false alarms in the crash set.

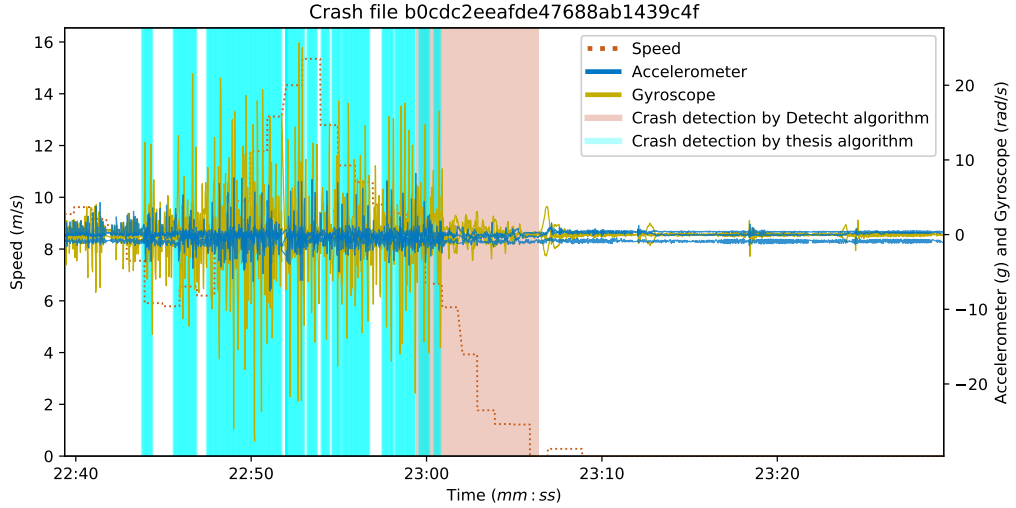


Figure 4.10: Zoom-in example of a lot of false alarms in the crash set.

4.1.5 Comparison of Detection Phases

Adding contextual attributes greatly helps reduce the number of false alarms, but can also decrease the capacity of detecting true crash events if the thresholds are improper. It is found by experiments that all these defined parameters affect the performance of the proposed algorithm, and increasing their thresholds can decrease both the FPR and TPR, which will be discussed in section 5.1.1. Note that due to different triggering logic of anomaly, the thresholds found in this thesis is different

from that of Detecht's current algorithm.

4.2 Supplementary Experiments

4.2.1 Mobile-AE with Different Length of Time Window

It is interesting to explore if CNN with a longer time window (more than 1000 ms) can better capture the crash dynamics in the temporal dependency. An additional MobileNetV3-small style encoder plus vanilla FCN decoder with a time window of 1280 ms are successfully trained though it was extremely slow to train the large model and predict the hundreds millions of samples. Fig 4.11 shows the distribution of reconstruction error of mobile-AE model with a longer time window (1280 ms) (mobile-AE-1280) trained on a larger subset of normal logfiles. It shows a similar observation with Fig 4.1. As for the threshold of reconstruction error to define an anomaly event, similar to the case in section 4.1.1, TH_A = 95% percentile is adopted in mobile-AE-1280. Due to the longer time window, it needs 2490 ms to raise a crash alarm, which is longer than that in the main results, but with a much higher FPR (shown in table 4.2) than that of optimal mobile-AE-300. Besides, all the sub-results are without speed cancellations.

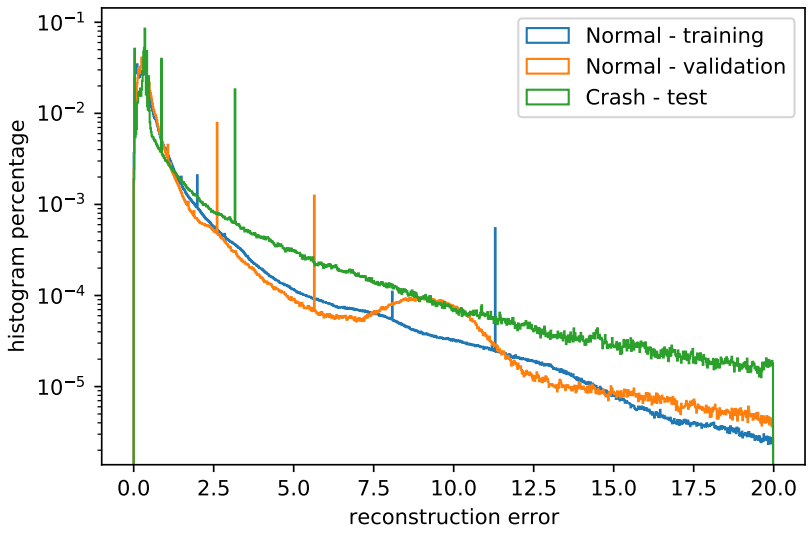


Figure 4.11: Distribution of reconstruction error of mobile-AE trained on the larger subset of normal logfiles.

4.2.2 VGG-AE Trained on the Larger Subset of Normal Logfiles

Though MobileNet-V3-small style CNN is specially designed for mobile applications, its computation cost still accounts for hundred thousands of parameters. To find if a simpler model can achieve the same performance in the MCC detection appli-

Evaluation Metrics	Crash Logfiles	Normal Logfiles
Specificity	96.74%	99.611%
False Positive Rate	3.26%	0.389%

Table 4.2: Specificity and false positive rate in crash logfiles and normal drivings in sub-results of mobile-AE trained on the larger subset of normal logfiles. It corresponds to one false alarm per 0.48 driving hours on average.

cations, a VGGNet style CNN architecture with much few trainable parameters is trained on the same subset of the normal logfiles in section 4.1.1. Several VGG style encoders plus vanilla FCN decoders with window length 1280 ms are tested but they all failed in training, so from now on all the VGG-AE models are with a time window of 300 ms. Besides, if not stated, the left results will aim at presenting the optimal parameters that can detect all the true crash events while maintaining the lowest FPR. T_SPEED will start from a certain threshold and then decrease by step, and T_ACC, T_GYRO are chosen from a list of candidate pairs.

Fig 4.12 shows the distribution of reconstruction error of this VGG-AE model trained on the larger subset of normal logfiles. It shows a similar observation with Fig 4.1 that the distributions in the three datasets follow the positively skewed distribution with the majority of reconstruction error focusing on small values, as well as that the crash set (test set) shows a high portion in larger reconstruction errors. Both the FPR and the statistics on one false alarm per driving hours are worse than that of the main results.

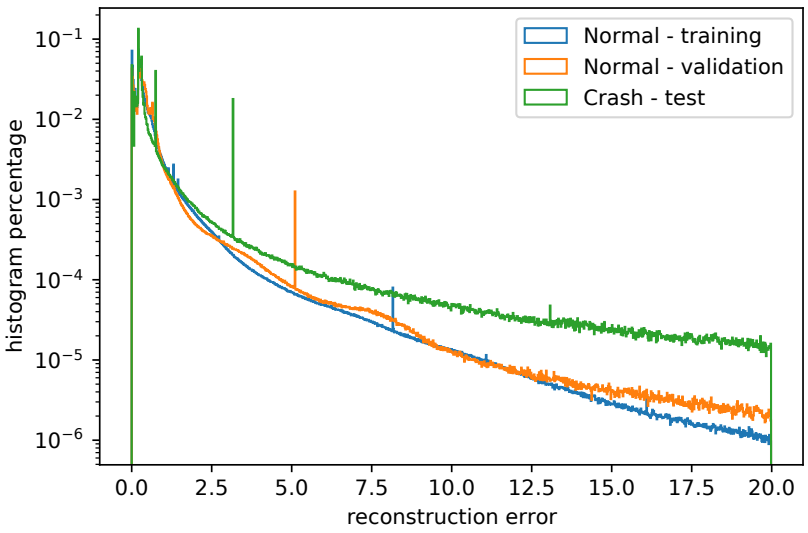


Figure 4.12: Distribution of reconstruction error of VGG-AE trained on the larger subset of normal logfiles.

In this model, setting TH_A as the 99% percentile of reconstruction error is not ideal because 2 real crashes cannot be detected even the TH_B is set 1, which represents point anomaly. Therefore, TH_A is set 98% percentile to achieve the sensitivity as 100% as detecting all the true crashes. Here the specificity corresponding to the tuned parameters is in table 4.3.

Evaluation Metrics	Crash Logfiles	Normal Logfiles
Specificity	98.66%	99.974%
False Positive Rate	1.34%	0.026%

Table 4.3: Specificity and false positive rate in crash logfiles and normal drivings in sub-results of VGG-AE trained on larger subset of normal logfiles. It corresponds to one false alarm per 0.67 driving hours on average.

4.2.3 VGG-AE Trained on the Smaller Subset of Normal Logfiles

There are hundred millions of samples provided by Detecht, thus training on the whole dataset would be very slow and impractical. Comparing with the famous ImageNet dataset (ILSVRC2012), it takes several days to train on the full ImageNet data with more than one million samples on a single GPU, or a couple of hours on hundreds of GPUs, let alone hundred millions of samples. Therefore, to investigate if training on the small subset of normal logfiles can achieve the same performance, these sub-results are based on the same VGG-AE model structure with previous section but trained on the smaller subset of the normal logfiles, with randomly chosen samples in each logfile. In short, the validation set in this part is the training set in the previous section and vice versa. The statistics of normal drivings are still based on the larger subset of normal logfiles.

The histogram distribution of the reconstruction errors is drawn in Fig 4.13. The difference is that in Fig 4.12 when trained on the larger subset of normal logfiles, the distributions of residual errors in the training and validation sets are more similar, meaning the model in Fig 4.12 does better generalize its learned normal patterns on the smaller subset of normal logfiles, compared with the model trained on the smaller subset as in Fig 4.13.

Evaluation Metrics	Crash Logfiles	Normal Logfiles
Specificity	98.89%	99.984%
False Positive Rate	1.11%	0.016%

Table 4.4: Specificity and false positive rate in crash logfiles and normal drivings in sub-results of VGG-AE trained on the smaller subset of normal logfiles.

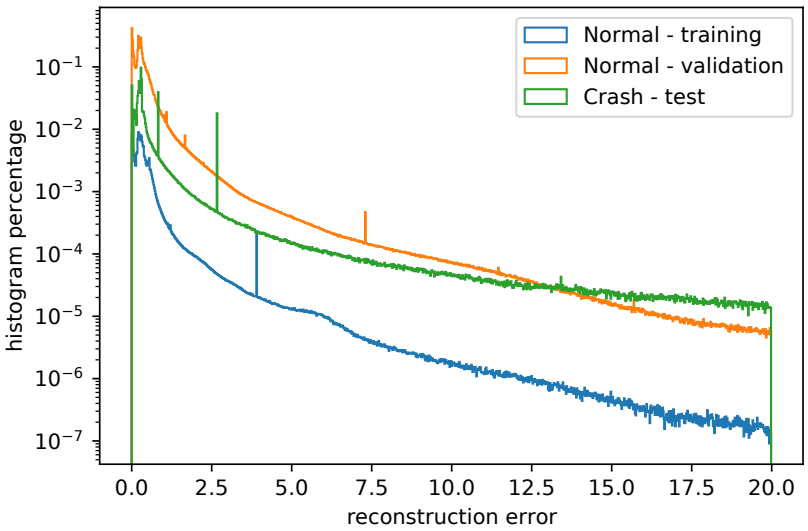


Figure 4.13: Distribution of reconstruction error of VGG-AE trained on the smaller subset of normal logfiles.

4.2.4 VGG-AE with Different Crash Detection Thresholds

The purpose of this part is to test if the false alarms can be reduced by adjusting the thresholds of the proposed algorithm. The VGG-AE model is the same as in the sub-results section 4.2.3. Table 4.5 is the corresponding specificity. This result shows that by increasing the threshold TH_B, the consecutive number of point anomalies that raises a collective anomaly, the false positive rate can be greatly suppressed, with the cost of decreasing sensitivity. With the mentioned parameters which only consider the length of collective anomaly and the speed constraint, only 8 true crash events are correctly detected. Moreover, by introducing the constraints of acceleration and gyroscope, the number of true positives will further decrease, but the number of average driving hours within one false alarm can be greatly increased.

Evaluation Metrics	Crash Logfiles	Normal Logfiles
Specificity	99.56%	99.994%
False Positive Rate	0.44%	0.006%

Table 4.5: Specificity and false positive rate in crash logfiles and normal drivings in the sub-results of VGG-AE.

Comparing these above-mentioned tables together, it is easy to find out that the grid search of TH_A and TH_B is reasonable: the set TH_A can capture all the true crash events correctly with corresponding other parameters and resulting in lower false positive rate, and the VGG-AE model with higher TH_A will resulting in a shorter length of crash samples (smaller TH_B) to raise a crash alarm. If both TH_A and TH_B are set too high, the model will fail in finding true crashes and pretend to predict all the input samples as normal, which is not expected working way. A

future tuning direction can be a comprehensive tuning combining all these hyper-parameters together, with the cost of lengthy simulations. In conclusion, focusing on false positive rate or specificity alone is meaningless, because the model is expected to detect real crash event to save the lives of motorcyclists while reducing false alarms.

5

Discussion

5.1 The Performance of MCC Detectors

The main results produced by mobile-AE-300 are the best one in concern of specificity with unit sensitivity compared with all the other sub-results. It is widely regarded that the model trained on a larger dataset is generally better than that on a smaller dataset because the larger dataset can cover more samples to let the sample distribution more approximate the population distribution [24]. However, the sub-results presented in this thesis may raise the concern of the training set: the larger training set may contain noises that deviate significantly from the normal driving patterns. This hypothesis can be supported by this phenomenon: all the models have higher specificity on the smaller subset of normal logfiles, no matter if they are trained on a larger subset or a smaller subset. In short, data cleaning shall be paid attention in order to train a better model, and some other methods can be done to handle the ill-behaved logfiles. When implementing the proposed algorithm in real-life scenarios, all the datasets of normal logfiles excluding the abnormal ones can be used to train the model for a better generalization purpose.

In addition, the distribution histogram of the reconstruction error among these models may further suggest an improvement for the model training. For the VGG-AE, it is found that the 98% percentile of reconstruction error in the sub-results trained on the larger normal subset is already higher than the 99% percentile of the VGG-AE model trained on a smaller normal subset, let alone 99% percentile, meaning this model trained on a smaller subset does better capture the normal driving behaviors compared with the previous model training on the larger normal subset which shows the larger variability in the larger training set. In short, a better method for splitting datasets will be stratified splitting, that the different driving recordings of the same motorcyclists are stratified sampled into the training set and validation set if such data is available, compared with the current simple sampling method. Compared with the presented results, all these models have the same capacity to identify all the true crashes, supporting the hypothesis of the AE methods that AE cannot well reconstruct the crash samples, but both VGG-AE models perform worse from the aspect of the specificity.

As for the simulated and real-world crashes, all the models can identify the crash events, but the model produced much fewer false alarms on simulated crash logfiles with while for the real-world crash data, more false alarms are raised. It might also

be due to the data cleanliness: the simulated crashes are carried out by Detecht with one iPhone, which owns better signal resolution and stability regarding sensor quality, while the real-world crashes are collected from various users with versatile smartphone models, thus the data quality cannot be guaranteed. It also raises the attention for data cleaning: if the incoming data quality is not guaranteed, the machine learning algorithm may not work as expected, and additional measures have to notify the users and ask them to keep the phone mounted/carried in a recommended way. For example, personalized feedback can be given to users before they risk raising false alarms. For example, the smartphone may not be in a tight placement when the speed is near 0 but the acceleration or gyroscope indicates that the user is moving. It might also help if an "inappropriate mounting detector" can be trained by learning the patterns when the speed is near 0.

Both with a same window length (300 ms) and trained on the same larger subset of normal logfiles, VGG-AE-300 and mobile-AE-300 have comparable performance, indicating the model architecture does not greatly influence the learning of hidden normal patterns when the models are relatively simple. It may indicate that trying other CNN architectures may be of no help in this case. Besides, the hidden dimension of mobile-AE is slightly higher than that of VGG-AE, so increasing the dimension of latent space may also help the MCC detection, which should be thoughtfully tuned. However, when the model structure becomes complex, the VGG-AE-1280 failed in learning a representative model but mobile-AE-1280 did. It may suggest that when designing the model architecture for more complex cases, the model structure should be carefully considered. Currently, the model structures are manually designed, while searching for an optimal model structure may be in a future direction. Besides, the decoders used in this thesis are all vanilla fully convolutional neural networks, whose performance may be suppressed by other more advanced CNN structures, or even an LSTM decoder.

5.1.1 Trade-off Between Crash Detection and False Alarms

When tuning the model hyper-parameters, there is always a trade-off between crash detection capacity and false alarm rate (sensitivity and specificity). When increasing the values of each threshold, the false alarms did decrease but so did the true alarms. An intuitive hypothesis for such a phenomenon is the separability of normal patterns and crash patterns. Using only the information from the accelerator and gyroscope, it might be hard to distinguish the crash and non-crash by a simple thresholding reconstruction method. In this case, more information may be needed to classify them more correctly, such as multi-head CNN model [32]. It might also be the case that, it might be hard for the autoencoder methods to solve this contradiction by learning the generic normality features from the abnormal training data [24]. Besides, a common usage of autoencoder is to make use of the latent space, so another direction may be to design some statistical or frequency features, and combining these features into a one-class classifier, rather than by the reconstruction error thresholding method [3].

In addition to the perspectives of the model algorithm, dataset can also be enhanced to solve this problem. There are two sides in the dataset view. One view is the possible separability of the slight and severe crashes. The assumption is that all the crashes, no matter of the severity, share the common patterns separate from normal patterns, while the distinction between severity may come from a threshold of crash dynamics: when one more second or a little bit heavier the motorcycle rotates, the severe crash happens. In this view, the model can eliminate the slight crashes if there are labeled slight crash data. The other view is to acquire more labeled crash data, to make the semi-supervised or unsupervised anomaly detection problem to the fully supervised classification problem, as investigated in [19, 20], because classification problem is much easier than anomaly detection problem. However, the collection of crash patterns can be tremendously expensive and costly, and collecting as many as different crash patterns may be unrealistic comparing with collecting different normal driving patterns.

5.1.2 Two Phases of Crash Detection

In the proposed algorithm there are two phases of crash detection. Phase one is the trigger of acceleration, angular velocity, and estimated speed derived from the GPS signal, while phase two is sending these information into neural networks for classifying whether a crash happens. The purpose of the two phases comes from two considerations. On one side, computation through neural networks (tons of matrix multiplication) can be power costly if the smartphone does not have a neural processing unit (NPU) dedicated for deep learning while computing if the sensor values exceeding a threshold is much easier. Secondly, when the sensor values are rather small but their patterns show abnormal, a simple thresholding methods can erase these false alarms detected by crash pattern recognition. In short, proper thresholds can greatly lessen the false positive rates.

5.1.3 The Length of Time Window

The crash may last for only milliseconds or longer, depending on the types of crashes. For example, hitting and being stuck on a tree may immediately stop the vehicle from moving within 1 second, while roll over or sliding crash may last longer. This thesis initialized the model structure with a time window 300 ms, based on the previous work by [10]. The inherent limitation of CNN-based temporal anomaly detection is the fixed time window: the CNN model can only learn the temporal dependency within the fixed time window. That's the reason why mobile-AE-1280 is proposed to learn the longer driving dynamics, though failed in achieving better performance. Exploring various CNN structures with different lengths of time window may be impractical because of the following reasons. Firstly, each length of time window needs a new model, and it takes time to train, optimize, and evaluate each model. Secondly, the designed model may be hard to train. Thirdly, the more complex the model structure is, the longer training and inference time it will take. Therefore, when the workload of CNN models is too large, other model

architecture specifically designed for temporal data, such as RNN, can be explored in future works. It should be noted that it is more troublesome and error-prone to train and tune temporal architectures, such as LSTM. Besides, it is also reported in [27] that LSTM only outperforms CNN in the case where long-range temporal dependencies exist. It is also found that GRU can produce similar results and simplify the LSTM architecture. [33] recommended adopting complex models when simple time-window-based methods are not performing as well as expected, similar to the principle of Occam's razor in machine learning.

5.2 Time Series Data Cleaning

This thesis shows the possibility that it is feasible to detect the crash events by thresholding the reconstruction error from autoencoders which learned the generic normality features. The result shows that the proposed algorithm well learns the normal driving patterns but it still raises false positives, which are a disturbing issue, no matter in the normal driving training set or the crash events test set. This phenomenon suggests that the alarms produced by the thesis algorithm is not simply a crash detector, but rather an outlier detector. By visually inspecting these false alarms in section 4.1.3 and 4.1.4, it shows that these false alarms often come with abnormal driving patterns, which may come from abnormal sensor measurement, smartphone loosely placing in the backpacks, and so on. Their high reconstruction errors suggest that the algorithm has not well learned the abnormal patterns. Together with the different performance of the main result and sub-results where the same models are trained on the different subset of normal driving logfiles, data cleaning is raised as a necessary step for further algorithm development, to only raise crash anomaly.

5.3 Limitations

There are certain limitations of the proposed algorithm. Firstly, the false positives are not eliminated in this thesis. Secondly, the tuned parameters are only a demonstration of this method, rather than optimal results. Thirdly, the interpretability of the model output is still in a black box. Fourthly, the real crashes detected by the proposed model last for only several milliseconds, where the crash dynamics cannot be recovered from the model output due to the lack of information in the available dataset. These limitations will be addressed in Chapter 7.

6
Conclusion

In this thesis, a subset of data from users of Detecht on motorcycle driving records was explored, which contains normal driving and crash logfiles. Literature review on MCC detection and anomaly detection was extensively studied. A convolutional autoencoder based anomaly detection method was developed, and different model architectures with different hyperparameters were studied for the performance of motorcycle crash detection. The results were compared and discussed from various perspectives. The undergoing crash events detected by the developed algorithm are partially overlapping with the crash start time defined by Detecht's current algorithm. Based on the main results and sub-results, CNN-AE is concluded to be an effective anomaly detector, but with some inherent limitations.

Regarding the aim of the thesis stated in section 1.3, the first and second aims have been conducted with promising results. In short, the major objectives have been accomplished, and more work can be further conducted based on the current thesis.

7
Future work

Due to the limitation of time and resources, not all interesting ideas have been implemented. Here are some suggestions for the future development of a better MCC detection algorithm based on the results of this thesis.

7.1 Data Collection

Precise labels are precious information. This idea is to convert an anomaly detection problem to a classification problem as the latter one is much easier. Currently, the Detecht's dataset only contains labels of normality and crash, while no more information is included in the normal driving data. It would benefit the data cleaning and model construction if it is known in data labels "nothing happened", "an MCC was close to happen", "a slight crash happened but I'm fine", "severe crash took place" similar to the concussion test proposed in [7].

Another idea is the mounting configuration of the smartphone. If the smartphone is mounted on the handlebar or the body of the motorcycle as expected, the crash detection can be much easier by focusing more on the driving patterns recognition and road profile (e.g., steer collision and rough road running), regardless of the smartphone motion dynamics coming from the human body movement or swinging in the loose backpacks. It is acceptable that the motorcycles cruise at the speed of 144 km/h, but unbelievable to see a motorcycle rotates at 10 rad/s.

7.2 Data Preprocessing

In this thesis, the input data to the machine learning model was only standardized to have zero mean and unit variance. It is found during the thesis experiments that proper standardization contributes to the convergence of autoencoder training. It is guessed that some other tricks may also enhance the model performance. Here the idea is that signal calibration techniques may improve the performance of the MCC detector. For example, [15] claims that real-time calibration of the IMU sensor information can be used to detect an MCC event. It is from the perspective of adding extra information during the driving. A similar idea is to add handcrafted features to the detector.

Transforming the time domain into the frequency domain may be of use to better capture the crash dynamics [14]. Besides, currently the sliding window is a rectan-

gular window, so will it help if other windows in classical digital signal processing, e.g., Chebyshev window [43], may be adopted?

Excluding the abnormal logfiles in the training set may also do a favor of reducing the false alarms. As discussed in section 5.1, models training on the larger dataset where more abnormal logfiles are presented have a higher FPR. From the point of this thesis, excluding the training logfiles which are predicted to have too many false positives and abnormal patterns may help improve the model performance by inspecting the complete logfiles, but it is not considerable or available in real-time MCC detection: we cannot know the future information before it does happen. One possible solution is to develop an anomaly detector for time series, which notifies the user the current IMU recording is not working properly and recommends some suggestions for an appropriate placement of the measuring device. For example, this anomaly detector can measure if any variable of the accelerometer and gyroscope often exceed a certain threshold within a fixed time window. If so, the crash detection algorithm can focus on contextual anomalies, without being extracted from confusing sensor recordings.

In addition, the stratified sampling can be applied to training set splitting, so that the model can learn the most useful features based on the most representative driving behaviors.

7.3 Outlier Detection

Section 5.2 raises the issue of data cleaning. From the academic view, the normal driving logfiles with abnormal recordings can be excluded from the training set to train a well-learned normality detector. In this sense, the trained detector can better recognize the normal drivings. However, the issue that the detector cannot distinguish crash patterns and abnormal patterns still exist. One possible direction is to extract and compare the crash and abnormal patterns from logfiles. Then some hints to tell the difference between them may be obtained from the corresponding analysis. Or to train another detector based on the extracted abnormal driving patterns, then it works as an outlier detector or false alarm reducer to cancel the false alarms raised by the thesis algorithm. Or to train a detector that can well learn both the normal and abnormal driving patterns based on the normal logfiles. For example, if an overcomplete autoencoder can do this? The advantage of this direction is that the outlier detection may learn the road anomalies, errors in sensor readings, or just meaningless random pose change in a loose backpack, which can be differentiated from real crash events.

Another direction of this issue is outlier repairing [44, 45]. Since errors are prevalent in sensor readings or the above-mentioned scenarios, one can recognize the dirty snippet first and then repair them. It is demonstrated to improve the time series classification performance in literature [45].

7.4 Dedicated Model Development

The concept of the contextual anomaly is adopted to the convolutional autoencoder in this thesis, but rather an adapted version of collective anomalies formed by sliding-window-based point anomalies. In the future, more exploration can be done in the hidden space of the autoencoder. For example, rather than feeding the raw sensor values into an RNN, the compressed features produced by the autoencoder may perform better acting as the input to RNN, a type of prediction model [2]. Multi-head neural networks may also be beneficial because it can separate heterogeneous sensor information without sharing the same feature extractor [32]. Ensemble learning by combining multiple models may also contribute [29]. Beyond the traditional deep anomaly detection methods, segmentation-based temporal anomaly detection is reported to work well [46]. Recently, neural architecture search (NAS) is quickly developing, which may help find the optimal model structure by automating designing deep learning architecture.

7.5 Optimal Implementation

In the current implementation, if two collective anomalies are separated by one point normality, the algorithm will produce two false alarms. Future implementation can concatenate the neighboring anomalies to reduce the number of false positive samples.

The model may be better trained with more advanced training options, such as one cycle learning rate policy. Besides, with selected training samples with considerable size, a more random and representative optimizer may help the learning go to a more flat local minimum, with better performance.

Hyperparameter tuning in this thesis is a grid search plus three short candidate lists. Quadrillion search considering five parameters at once may be the optimal tuning method if time and resources allows.

Bibliography

[1] Transport Analysis, "Road traffic injuries 2020," 2020, [accessed 19-May-2021]. [Online]. Available: https://www.trafa.se/en/road-traffic/road-traffic-injuries/

[2] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021, doi:10.1145/3439950.

[3] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–33, 2021, doi:10.1145/3444690.

[4] Harold de Bock, "Motorcycle safety and accidents in europe," 2016, [accessed 12-February-2021]. [Online]. Available: http://www.fema-online.eu/website/index.php/2016/08/05/motorcycle-safety-and-accidents/

[5] P. Puthan, N. Lubbe, J. Shaikh, B. Sui, and J. Davidsson, "Defining crash configurations for powered two-wheelers: comparing iso 13232 to recent in-depth crash data from germany, india and china," *Accident Analysis & Prevention*, vol. 151, p. 105957, 2021, doi: 10.1016/j.aap.2020.105957.

[6] United States Department of Transportation, "Motorcycle safety," 2018, [accessed 12-February-2021]. [Online]. Available: https://www.nhtsa.gov/road-safety/motorcycle-safety

[7] G. Matuszczyk and R. Åberg, "Smartphone based automatic incident detection algorithm and crash notification system for all-terrain vehicle drivers," Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2016.

[8] E. Commission, "Traffic safety basic facts 2018 – motorcycles mopeds," European Commission, Tech. Rep., 2018.

[9] Swedish Transport Administration, "Analysis of road safety trends 2018: Management by objectives for road safety work towards the 2020 interim targets," 2019, [accessed 24-February-2021; Figure 36, 46]. [Online]. Available: http://trafikverket.diva-portal.org/smash/record.jsf?dswid=2826&pid=diva2%3A1389250&c=7&searchType=SIMPLE&language=en&query=Analysis+of+Road+Safety+Trends&af=%5B%5D&aq=%5B%5B%5D%5D&aq2=%5B%5B%5D%5D&aqe=%5B%5D&noOfRows=50&sortOrder=author_sort_asc&sortOrder2=title_sort_asc&onlyFullText=false&sf=all

[10] S. A. MIAN, "Development of a smartphone-based crash notification system for motorcycle drivers using machine learning," Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2020.

[11] G. Gil, G. Savino, S. Piantini, N. Baldanzini, R. Happee, and M. Pierini, "Are automatic systems the future of motorcycle safety? a novel methodology to prioritize potential safety solutions based on their projected ef-

fectiveness," *Traffic injury prevention*, vol. 18, no. 8, pp. 877–885, 2017, doi:10.1080/15389588.2017.1326594.

[12] WHO, "Global status report on road safety 2018: Time for action," World Health Organization, Tech. Rep., 2018.

[13] European Commission, "Safety design needs and protective equipment for motorcycles," 2021, [accessed 07-April-2021]. [Online]. Available: https://ec.europa.eu/transport/road_safety/specialist/knowledge/vehicle/safety_design_needs/motorcycles_en

[14] S. Gelmini, G. Panzani, and S. Savaresi, "Analysis and development of an automatic ecall for motorcycles: a one-class cepstrum approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3025–3030, doi:10.1109/ITSC.2019.8916907.

[15] S. Gelmini, S. Strada, M. Tanelli, S. Savaresi, and C. D. Tommasi, "Automatic crash detection system for two-wheeled vehicles: design and experimental validation," *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 498–503, 2019, 9th IFAC Symposium on Advances in Automotive Control AAC 2019.

[16] S. Gelmini, S. C. Strada, M. Tanelli, S. M. Savaresi, and C. De Tommasi, "A novel crash detection algorithm for two-wheeled vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 88–99, 2021.

[17] Y. Kobayashi and T. Makabe, "Crash detection method for motorcycle airbag system with sensors on the front fork," in *23rd International Technical Conference on the Enhanced Safety of Vehicles*, 2013.

[18] D. Selmanaj, M. Corno, and S. M. Savaresi, "Hazard detection for motorcycles via accelerometers: A self-organizing map approach," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3609–3620, 2017, doi:10.1109/TCYB.2016.2573321.

[19] J. Parviainen, J. Colliri, T. Pihlström, J. Takala, K. Hanski, and A. Lumiaho, "Automatic crash detection for motor cycles," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, 2014, pp. 3409–3413, doi:10.1109/IECON.2014.7049003.

[20] F. Tabei, B. Askarian, and J. W. Chong, "Accident detection system for bicycle riders," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 878–885, 2021, doi:10.1109/JSEN.2020.3021652.

[21] J. Schnee, J. Stegmaier, and P. Li, "A probabilistic approach to online classification of bicycle crashes," *Accident Analysis Prevention*, vol. 160, p. 106311, 2021, doi:https://doi.org/10.1016/j.aap.2021.106311.

[22] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[23] L. Erhan, M. Ndubuaku, M. Di Mauro, W. Song, M. Chen, G. Fortino, O. Bagdasar, and A. Liotta, "Smart anomaly detection in sensor systems: A multi-perspective review," *Information Fusion*, 2020, doi:10.1016/j.inffus.2020.10.001.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[25] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, mod-

els, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018, doi:10.1016/j.inffus.2017.12.007.

[26] M. Farahani, "Anomaly detection on gas turbine time-series' data using deep lstm-autoencoder," Master's thesis, Umeå University, Umeå, Sweden, 2021.

[27] A. SINGH, "Anomaly detection for temporal data using long short-term memory (lstm)," Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2017.

[28] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, 2018, pp. 125–134, doi:10.1109/MDM.2018.00029.

[29] E. Principi, D. Rossetti, S. Squartini, and F. Piazza, "Unsupervised electric motor fault detection by using deep autoencoders," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 441–451, 2019, doi:10.1109/JAS.2019.1911393.

[30] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2020, doi:10.1109/JIOT.2019.2958185.

[31] W. Lu, Y. Li, Y. Cheng, D. Meng, B. Liang, and P. Zhou, "Early fault detection approach with deep architectures," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, no. 7, pp. 1679–1689, 2018, doi:10.1109/TIM.2018.2800978.

[32] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head cnn-rnn for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 2019, doi:10.1016/j.neucom.2019.07.034.

[33] J. S. Felix A. Gers, Douglas Eck, "Applying lstm to time series predictable through time-window approaches," in *Artificial Neural Networks — ICANN 2001*. Springer, 2001, pp. 193–200, doi:10.1007/3-540-44668-$0_9$3.

[34] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Anomaly detection using autoencoders in high performance computing systems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 9428–9433, Jul. 2019, doi:10.1609/aaai.v33i01.33019428.

[35] Apple, "Getting raw accelerometer events," 2021, [accessed 16-May-2021]. [Online]. Available: https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events

[36] J. Wu, L. Yao, B. Liu, Z. Ding, and L. Zhang, "Combining oc-svms with lstm for detecting anomalies in telemetry data with irregular intervals," *IEEE Access*, vol. 8, pp. 106 648–106 659, 2020, doi: 10.1109/ACCESS.2020.3000859.

[37] O. Karaahmetoglu, F. Ilhan, I. Balaban, and S. S. Kozat, "Unsupervised online anomaly detection on irregularly sampled or missing valued time-series data using lstm networks," *arXiv preprint arXiv:2005.12005*, 2020.

[38] A. Zhang, S. Song, J. Wang, and P. S. Yu, "Time series data cleaning: From anomaly detection to anomaly repairing," *Proceedings of the VLDB Endowment*, vol. 10, no. 10, pp. 1046–1057, 2017, doi:10.14778/3115404.3115410.

[39] C. Sun, S. Hong, M. Song, and H. Li, "A review of deep learning methods for irregularly sampled medical time series data," *arXiv preprint arXiv:2010.12493*, 2020.

[40] Detecht Techonologies AB, "Frequently asked questions," 2021, [Online; accessed 11-May-2021]. [Online]. Available: https://detecht.se/faq

[41] NVIDIA Corporation, "Convolutional layers user guide," 2021, [accessed 17-Aug-2021]. [Online]. Available: https://docs.nvidia.com/deeplearning/performance/dl-performance-convolutional/index.html#channels

[42] E. J. M. Jerone T. A. Andrews and L. D. Griffin, "Detecting anomalous data using auto-encoders," *Journal of Machine Learning and Computing*, vol. 6(1), pp. 21–26, 2016, doi:10.18178/ijmlc.2016.6.1.565.

[43] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. USA: Prentice Hall Press, 2009.

[44] X. Wang and C. Wang, "Time series data cleaning with regular and irregular time intervals," *arXiv e-prints*, pp. arXiv–2004, 2020.

[45] A. Zhang, S. Song, J. Wang, and P. S. Yu, "Time series data cleaning: From anomaly detection to anomaly repairing (technical report)," *arXiv preprint arXiv:2003.12396*, 2020.

[46] T. Wen and R. Keyes, "Time series anomaly detection using convolutional neural networks and transfer learning," *arXiv preprint arXiv:1905.13628*, 2019.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY