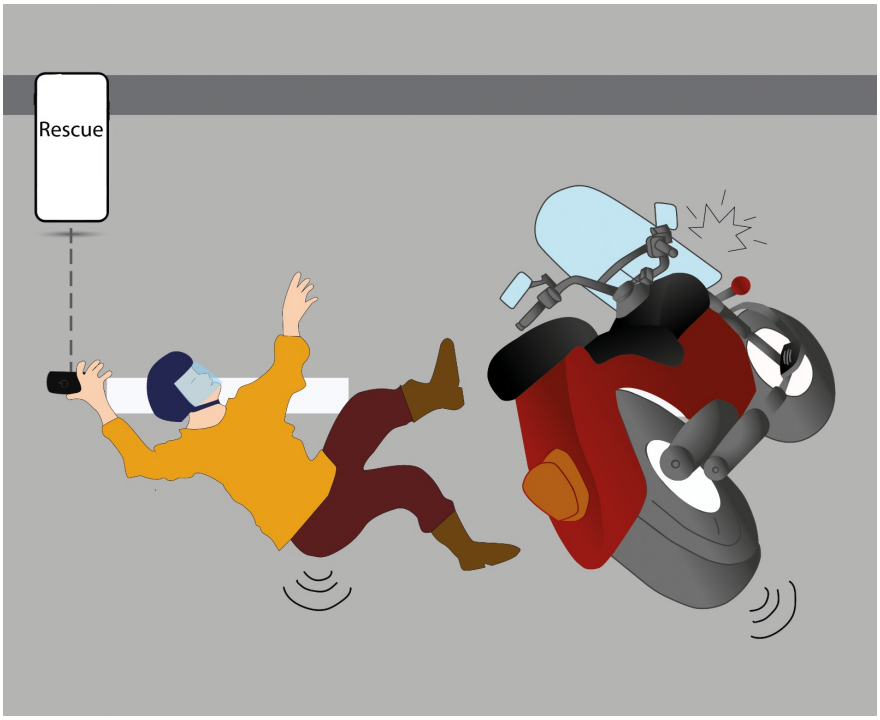
CHALMERS
UNIVERSITY OF TECHNOLOGY



Development of a Smartphone-based Crash Notification System for Motorcycle Drivers using Machine Learning

Master's thesis in Electrical Engineering

SAUD AHMAD MIAN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Master's thesis 2021

Development of a Smartphone-based Crash Notification System for Motorcycle Drivers using Machine Learning

Using Machine Learning Algorithm to Predict Motorcycle Crashes and Notify Alerts to Emergency Contacts .

SAUD AHMAD MIAN

CHALMERS
UNIVERSITY OF TECHNOLOGY

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

Development of a smartphone-based crash notification system for motorcycle drivers using machine learning.
SAUD AHMAD MIAN

Supervisor: Stefan Candefjord, Assistant Professor, Department of Electrical Engineering, Chalmers University of Technology
Examiner: Bengt Arne Sjöqvist, Professor of Practice emeritus, 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

Abstract

The absence of a vehicle shell around the motorcyclist makes the motorcycle crashes fatal as the rider collides directly with the objects. During motorcycle crashes, when the rider is driving alone, there is no one to call for emergency rescue and first aid. Moreover, when the rider is in the countryside or at a time or place where there is no one to help, there rises a need for an automatic emergency alert system for motorcycle riders. This report is about the thesis with Detecht motorcycle crash detection company. Developing a crash prediction algorithm that can be used with Detecht's smartphone-based application. It includes the crash prediction using a machine learning algorithm in python. Using the parameters from the sensors that are built-in in all smartphones nowadays. Eliminating the need for any separate device required for collecting the data needed to predict a crash and generate an alert message accordingly.

The crash samples in Detecht's data that includes the 26 simulated crashes and 3 real crashes are a small number of samples as compared to the 940214 samples of normal driving samples recorded from 76 Detecht's smartphone application users. It makes this crash prediction, an anomaly detection case. To have the best accuracy in results and to reduce the false alarms generated by the algorithm, different reasons for crashes are studied. These reasons show that a crash is not only a motorcyclist's mistake but there are many other factors involved in them.

Convolutional Autoencoder is the machine learning algorithm used for this anomaly detection, which is an artificial neural network. The algorithm works by reducing the dimensions of the data and regenerating the data using the learned reduce dimensions. Mean error distribution is used to evaluate the output of the algorithm by comparing the algorithm output to the actual data. This comparison predicts the crashes out of normal driving of the motorcyclists.

The report consists of a comparison of the convolutional autoencoder results with the previously used algorithm by Detecht and discuss problems using the machine learning algorithm to predict crashes. The behavior of the convolutional autoencoder is studied by relating the algorithm response to the respective sensor values at the time of the crash. A total of 85% of the 940214 data samples are predicted correctly having no false alarms and 62 % of the 26 simulated crashes were predicted correctly using the convolutional autoencoder. The results of the thesis emphasize more on reducing the false alarms from the algorithm. Since these false alarms can be irritating for the user which will result in users not trusting the application.

Keywords: Crash prediction, motorcycle crashes, anomaly detecion, machine learning, convolutional autoencoder, python.

Acknowledgements

As the thesis work progressed certain people provided help at different stages, Data used for developing the algorithm was provided by Niklas Ohlsson from Detecht. Detecht provided its kind permission to use the data for this thesis. Linh Tuan Nguyen has already worked on analyzing Detecht's data. His reports helped to understand the behavior of the data. Thesis supervisor Stefan Candefjord provided his guidance from the start keeping the work going in a good direction, giving feedbacks and solutions to the problems raised during the thesis. Stefan Candefjord not only provided useful links to the articles throughout the thesis which helped me understanding the problems I faced but also kept on motivating me to complete tasks. This improved my programming skills in Python and experience with machine learning algorithms.

Saud Ahmad Mian, Gothenburg, February 2020

Contents

List of Figures

List of Tables

1
Introduction

Motorcycle crashes cause much more injuries to the rider as compared to any other vehicle crashes. Moreover, the fatality rate of motorcycle crashes is higher as compared to car crashes [1]. A vehicle provides better active and passive safety features like seat belts and airbags etc. The motorcyclists are more likely to hit the objects directly in case of an accident, as they are not in a protective structure. On the contrary, the body of the car acts as a shell to capsule the passengers inside its body, protecting car passengers from direct contact with other objects during crashes.

Mostly the rider is alone when these crashes occur. There is no one to call for emergency rescue and first aid. If there is a crash in a city, there are people to help or call for an emergency in good time. But when the rider is in the countryside or at a time when there is no or low traffic, there rises a need for an emergency alert system for motorcycle riders. Many lives can be saved by giving proper first aid care at the right time. In this era, when everyone has smartphones, this alert system can be made as an application in a smartphone that can detect crashes and generate alerts for rescue, which is the concept of this thesis. To develop a machine learning algorithm using some programming language that can predict a crash. This prediction could be used by the company Detecht in their android and iOS application for generating alert messages or emergency calls in case if a motorcycle crash is predicted.

1.1 Reasons of Motorcycle Crashes

Developing an algorithm requires the study of the different causes of the problem. This thesis includes the study of the reasons that cause motorcycle crashes. There can be many reasons for a motorcycle crash, which include accidents at junctions, failure to control over bends, overtaking, loss of control, high speed, or use of alcohol. RoSPA Road Safety Research published a report [1] about common causes of motorcycle accidents in 2017. According to the RoSPA report, the number of motorcycle accidents over bends is 33% more than car accidents over bends, because the motorcycle is not much stable especially in bad weather conditions. These accidents are mostly because of the error of the rider, when the speed to maintain over the bend is misjudged. More than half of motorcycle accidents occur at junctions when the car drivers approach the path of an oncoming motorcycle without noticing them. These are mainly because of the car driver's fault [2]. Car drivers fail to see motorcyclists and come in front of them having much slower speed as compared to

the motorcyclist.
The third common reason is overtaking collisions. These are more common among young riders [1]. They occur when the rider is moving through the slow-moving traffic and gets hit by a car trying to change lanes. These collisions also take place when riders are changing lanes at high-speed, known as zic-zac driving. The rider fails to see a slow-moving vehicle hidden in front of some heavy vehicle while changing the lane and collides with it.
Another reason that causes accidents is the loss of control. These are very important to study since the majority of these accidents are related to the rider's error and slippery surfaces. In some cases, loss of control crash is the reason in addition to high speed, and use of alcohol cases [3]. Accidents are also caused by exceeding the speed limits. However, this is more common with motorcycles having an engine size of more than 500cc involved in a crash. For the cases involving motorcycles having smaller engines or the cases which are not related to high-speed driving, factors of careless and reckless driving are commonly involved [2].
The lastly discussed reason in the report is the use of alcohol. This decreases the concentration and reaction time of the rider resulting in misjudging the situation and causing an accident. The percentages of these different reasons for crashes are in the table 1.1.

Reasons of motorcycle crashes	Percentage of crashes
Collision at junctions	64
Alcohol	39
Loss of control	20
Overtaking collision	15
Rear-end collision	11
High speed	9.2
Failure to negotiate bends	9

Table 1.1: Percentage of different reasons of motorcycle crashes from ACEM

The table 1.1 is formed using the information from the different studies of the Association of European Motorcycle Manufacturers (ACEM). ACEM performed Motorcycle Accidents In-Depth Studies knows as MAIDS [4], comparing the causes of motorcycle crashes. For this purpose, approximately 800-900 motorcycle crashes were studied. According to ACEM reports, most of the crashes were at the junctions hit by other vehicles. It is a general observation from the MAIDS study that there is not always one single reason for a crash. Therefore, most of the crashes can fit in even two or three reasons in the above table, making the data overlapping in percentages.

1.1.1 Accidents Comparison

With the growing number of traffic on roads, the risk of having traffic accidents also increases [5]. Although every car manufacturer company is struggling to provide better safety features, there are not many safety features that can be designed for motorbikes. Even a rider having a helmet and proper safety kit can have serious injuries [1]. The motorcycle accidents cannot be linked only with riding carelessly or riding at high-speeds [3]. As the report [1] shows that in some accidents, the rider is not even making any error, driving carefully following all traffic rules and get hit by a careless car driver. Or the rider can lose grip on the road because of weather or road conditions. Sometimes rider loses their balance in attempting to save any person or animal crossing the roads at distances where carefully stopping the motorcycle is not possible. Since there are few motorcycles on roads as compared to cars and other vehicles therefore their accidents are not enormous in numbers as compared to everyday car crashes. But the ratio between the fatalities caused by motorcycle accidents is increasing as compared to motorcar accidents fatalities. Because more and more safety features are introduced in cars, making them safe to drive. The figure 1.1 is the graph from data collected by the National Highway Traffic Safety Administration (NHTSA) in 2010 [2]. This shows that how motorcycle accidents are increasing every year in the USA, which highlights the requirements for safety features for motorcycle users.

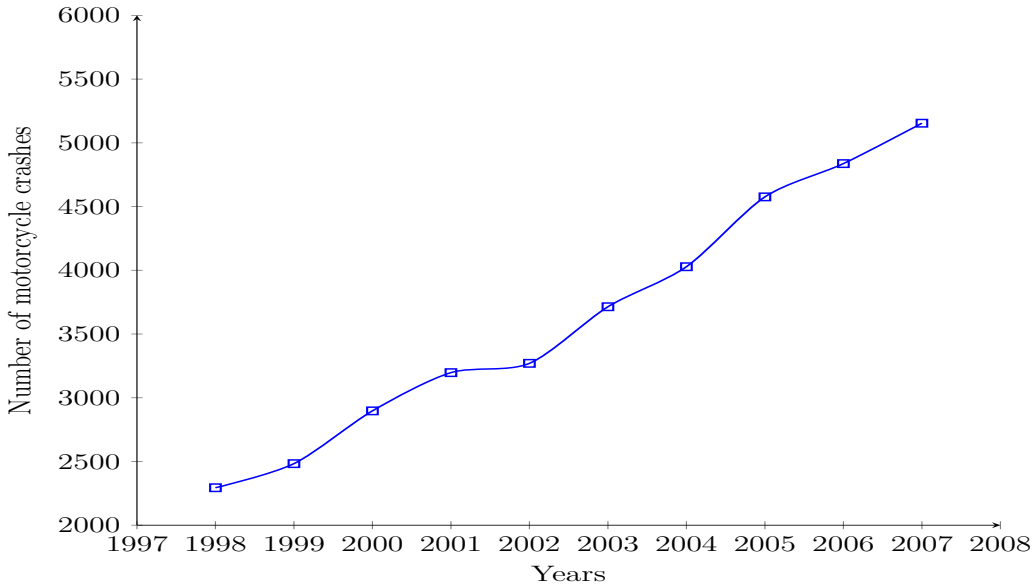


Figure 1.1: Graph of motorcycle accidents from NHTSA data

In comparison to the increasing motorcycle accidents shown in above figure, it is good to compare vehicles including cars, lorries and trucks etc which got involved in accidents during the same years, shown in figure 1.2. This shows that beside increasing in number of vehicles every year, their accident ratio is decreasing every year. This is mainly because of the increasing safety measures for vehicles.

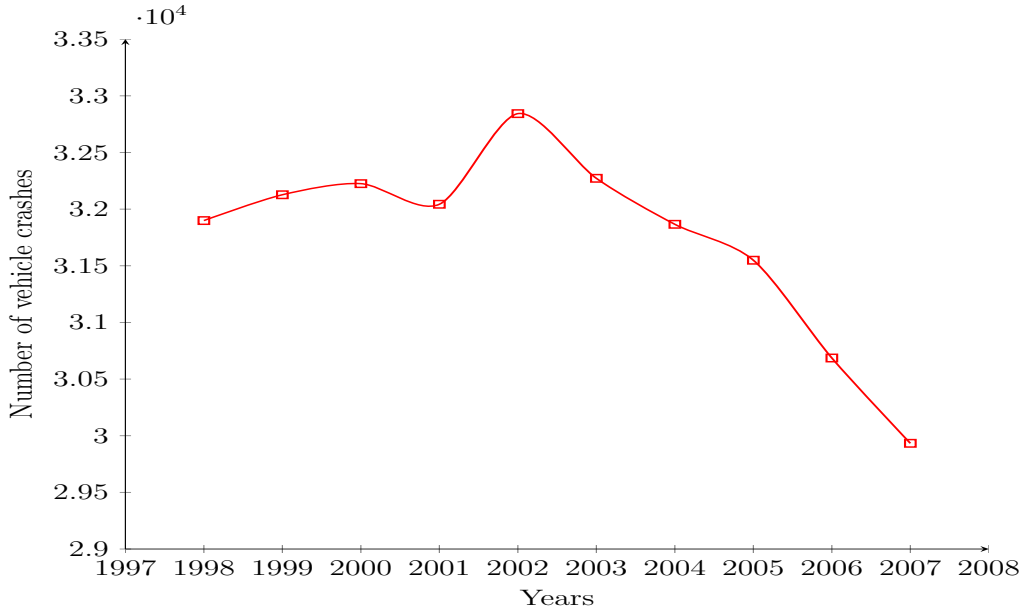


Figure 1.2: Graph of cars/ trucks accidents from NHTSA data

The comparison between these two graphs 1.1 and 1.2, highlights the importance of safety systems to be developed for motorcycles.

1.2 Purpose

With the increasing number of new and young motorcyclists, the risk of motorcycle crashes also increases. Inherently motorcycles are more susceptible to accidents due to their poor stability and high center of gravity. Most rider's lives can be saved from motorcycle accidents by giving proper first aid at the right time after the crash [6]. There are motorcyclists who love to cruise driving away from the cities and traffic. For such riders getting into any crash, even small injuries can cause causalities if not timely rescued and treated [7]. For this and many other causes of accidents stated above, having an automatic emergency alert system for motorcycle riders is necessary.
This thesis develops and evaluates a machine learning algorithm using the python programming language to predict motorcycle crashes. Almost everyone has a smartphone nowadays. Therefore, it is practical that motorcyclists use their phones for this system rather than having a separate device for this purpose. As a crash alert app on their smartphones can fulfill the requirements of data needed for prediction. The use of GPS for maps is very common for riders. Phones also have the accelerometer and gyro sensors built-in in them. These sensors from smartphones will be used for crash detection. The accuracy of these predictions is very important since false predictions will generate false alarms or the system not working properly can miss a serious accident alert.

1.3 Aim

The primary aim of this thesis is to:

- Study different reasons for motorcycle crashes.
- Develop a machine learning based crash detection algorithm from naturalistic data collected by Detecht's users and data from real and simulated crashes.
- Compare the machine learning algorithm accuracy and flexibility to Detecht's current algorithm.
- Reducing the false alarms when there is no crash, these can help develop users's trust in the app and letting them use it all the time.

1.4 Scope and Limitations

The data is collected using the Detecht app running on smartphones from different brands and years. The sensors of different smartphones are not equally calibrated. For the same jerk, two different smartphones can record the different magnitudes of values from the sensors. Similarly, the position of keeping the smartphone in a pocket or in a bag can add errors in the recorded values. The speed of the motorcycle calculated from GPS values is not accurate and has errors because the GPS values update after an interval in smartphones. This adds a range of errors in the location that can be up to a few meters in distance. Therefore it cannot detect sudden changes in the speed of moving motorcycles. This limits the Algorithm to focus more on accelerometer and gyro sensor values.

2
Theory

This section includes the explanation of the methods and the theoretical background which can help understand the working of the project.

2.1 Machine Learning

Machine learning is a branch of artificial intelligence that is nowadays a powerful tool to automate the process of data analysis. It minimizes human intervention in the process by learning and improving from the results. It uses computer capabilities to predict the results. An algorithm is defined or a predefined algorithm is used to compute data. Machine learning classifies the data into different classes, but in this thesis, the data will be separated into two classes. The possible outcome can be predicted motorcycle crash or predicting that it is no crash. Python language is used for programming as it is open-source and provides very useful functionality and libraries for machine learning that reduces the development time.

2.1.1 Anomaly Detection

Since there are very few accidents or crashes occurring as compared to the number of riders driving normally everyday. This will be a problem for balancing classes as if the data is classified into two classes crash or no crash, crash data will be significantly small as compared to no crash data. Since crash can be an event of a few seconds duration in the data log of a rider which can be hours long in time duration driving normally everyday. Which makes it an anomaly detection problem and not a classification problem. Anomaly detection is predicting the outliers or novelties or rare events and separating them from the majority of data. This gathers all the data that has almost the same qualities and properties and then the rare occurring events that are odd ones are separated from them.
There are many examples of anomaly detection. To start working on the data provided by Detecht, a similar problem of anomaly detection of credit card fraud detection [8] was examined. This data had very few frauds and can be related to the crash detection problem for learning purposes.

2.1.2 Evaluation of Predictions

When predicting the results during data analysis, it is very important to know how close are the predicted values to the actual defined values. This is known as accuracy

and it helps to understand and improve the correctness of results. The relation for accuracy is given by equation 2.1

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{2.1}$$

where: TP = True positive; FP = False positive; TN = True negative; FN = False negative

There are some other parameters commonly used to evaluate the results, to know how many positive predictions are actually true and are not just false alarms. This is called sensitivity and it is an important evaluation parameter in this thesis. The higher sensitivity means the maximum number of motorcycle crashes are correctly predicted. But it also increases the probability of higher false alarm rates or false crash predictions. These false alarms may cause disturbance for the users and their trust in the alert application will reduce. However, keeping the sensitivity maximum is important to predict all possible crashes. The equation 2.2 below gives the sensitivity of the predictions.

$$Sensitivity = (TN)/(TN + FP) \tag{2.2}$$

Another parameter is to correctly determine the rate of negative predictions. In this thesis, the negative prediction is no crash, and the rate of this happening and then correctly predicted by the algorithm is known as Specificity. The specificity is given by the equation 2.3.

$$Specificity = (TP)/(TP + FN) \tag{2.3}$$

The relation between these two parameters, sensitivity and specificity can be explained by two cases graphically presented below.

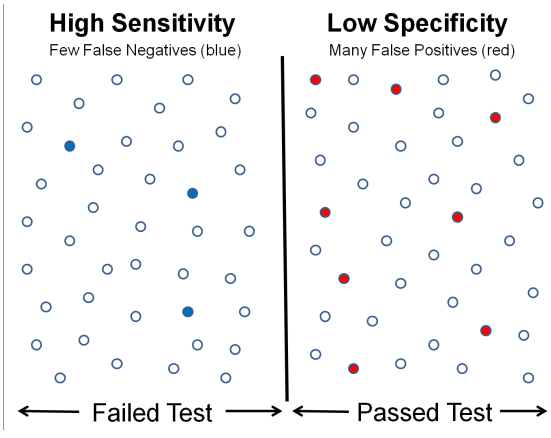


Figure 2.1: Sensitivity and Specificity: Case 1
Source: https://wikipedia.org/wiki/Sensitivity_and_specificity

In figure 2.1, blue dots represent false-negative rate and red dots represent false-positive rate. The lower the number of blue dots shows high sensitivity but it will increase the number of red dots which represents a false-positive rate and hence having low specificity. This is a trade-off between these two quantities. Some predictions require high sensitivity others require high specificity. Like in this crash prediction application, it is very important that every crash is predicted correctly and not a single crash is missed. But on the other hand, no rider wants false alarms or unnecessarily alert messages which will automatically be generated when the sensitivity is very low. This case is represented in figure 2.2 below.

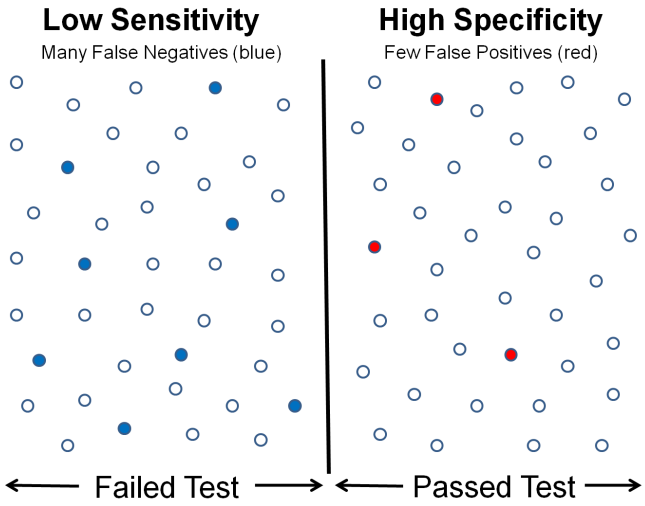


Figure 2.2: Sensitivity and Specificity: Case 2
Source: https://wikipedia.org/wiki/Sensitivity_and_specificity

The figure 2.2 represents the second case where the false-negative rate represented by blue dots is higher resulting in low sensitivity. But the red dots in this case representing false-positive rate are much lower having high specificity. These evaluation parameters are very important to understand the classification problems.

2.2 Imbalanced Data

The first approach to deal with a classification problem is to analyze the data. This process is called Exploratory Data Analysis (EDA) [9]. In exploring the data and labeling it in different classes sometimes the distribution of data in one class is huge as compared to the other class. This unequal distribution makes it difficult to draw a line between the elements of these classes as they are not distributed in an equal ratio. This is the case of imbalanced data [10]. On the other hand, the balanced data sets have data equally distributed in two or more classes. A well-known example of balanced data set is the Breast Cancer data set [11] which has an approximately equal distribution of data in the classes cancer or no cancer having a ratio of 1:1. Figure 2.3 shows the equal and unequal distribution of data in balanced classes on the left chart and Imbalanced classes on the right chart respectively.

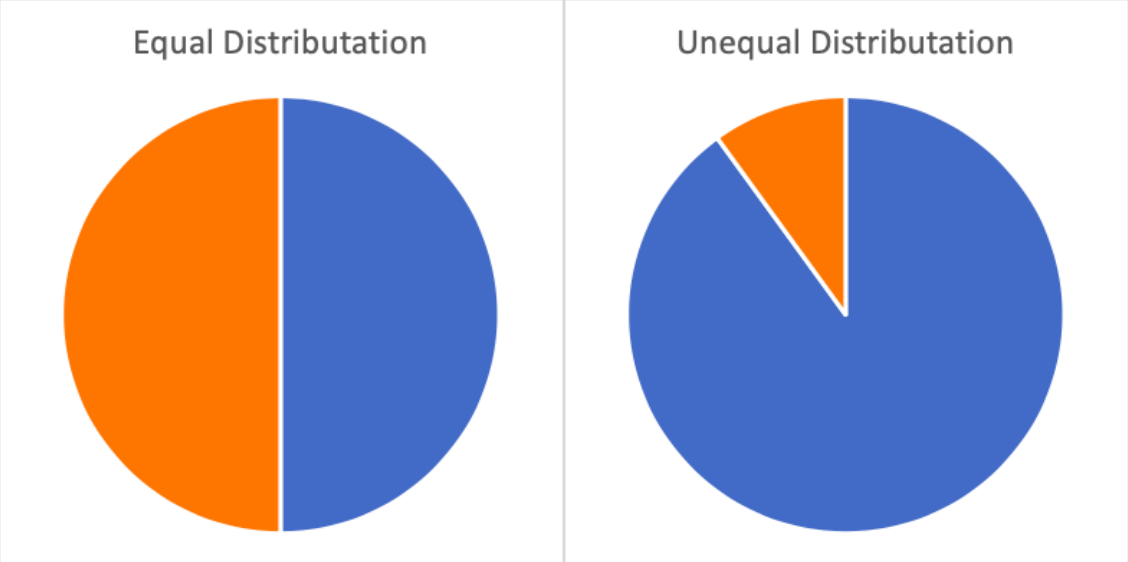


Figure 2.3: Balanced and Imbalanced Distribution

Anomaly class data sets also fall in the category of imbalanced data sets, but their distribution ratio in different classes has huge differences having very few data examples in one class called the anomaly class as compared to the normal class. Credit card fraud data set [8] is an example of imbalanced data set which is an anomaly detection problem where the fraud data class has very few entries as compared to no fraud data.

2.2.1 Techniques For Imbalanced Data

The two commonly used techniques to process the data creating equal distribution in different classes are undersampling and oversampling. In undersampling [12], some of the data points are randomly deleted from the class having majority data. This is done in order to match the number of data points in the class having only few examples. The other technique is oversampling which is a complicated process as compared to undersampling. In oversampling technique [13] random data samples are generated which have the attributes of the minority class. These data samples are added to the minority class to make this class having proportionally equal data points as of majority class. The figure below helps understanding this concept of undersampling and oversampling easily.

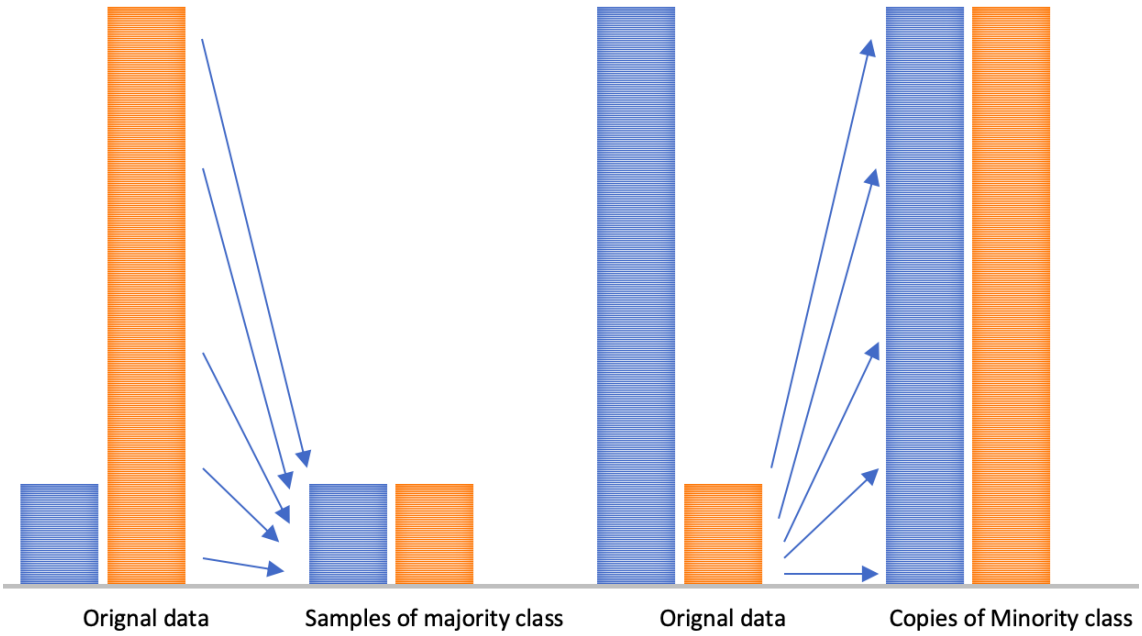


Figure 2.4: Undersampling And Oversampling

The graph on the left shows original imbalanced data, the majority class represented in orange is undersampled shown with arrows to meet the ratio of the minority class. Similarly, the two graphs at the right represent oversampling, the random copies of original data are created to meet the ratio of the majority class.

2.2.1.1 Ensemble of Sampler

In classification problems, sometimes the problem is too complex to be undersampled or oversampled using only one algorithm. The solution to this problem is to use multiple algorithms to estimate the results and improve performance which has effective results as compared to any algorithm used alone. These algorithms work as a system having democratic voting. It combines multiple estimators on a subset of data that is randomly selected from the major class. Bagging methods [14] is a popular example of this approach to such problems. There is an ensemble classifier known as bagging classifier [15] in the scikt-learn library. Using this classifier for imbalanced data, the estimators re-samples each subset of data and try to estimate the majority class as the equal ratio of the minority class.

2.3 One Class SVM

A supervised machine learning algorithm that separates one class from another class by learning the hyper-planes in the multidimensional data is known as Support Vector Machine (SVM) [16]. These algorithms are used to solve classification problems having multi-class data. However, one class SVM is used for classification problems where all the data available for training is of single class. SVM is trained to learn the patterns in this single class which is the normal class. The new data that is labeled outside this normal class is identified as anomaly data. In the sklearn library

in python language there is specific algorithm designed for such anomaly detection which is known as “novelty detection”.

2.4 Time Series Data Anomaly detection

Unlike the credit card fraud data, the data from Detecht which will be used in this project is time-dependent. Many time-series data are characterized by strong randomness and high noise [17]. For crash prediction using anomaly detection techniques, the events of crashes and no crashes should be selected to have complete information about the crash. This is also done in order to avoid false alarms. The figure below shows the representations of an outlier from the data in two different ways. The one On the left is independent of time and on the right is dependant on time. In the left figure, it is easier to identify the outlier as it is one point, but on the right, with the time axis, it is very difficult to classify the anomaly event because of time dependency making it complex to differentiate. The outlier is not just one peak, it is an event having information that has some peaks and some normal data.

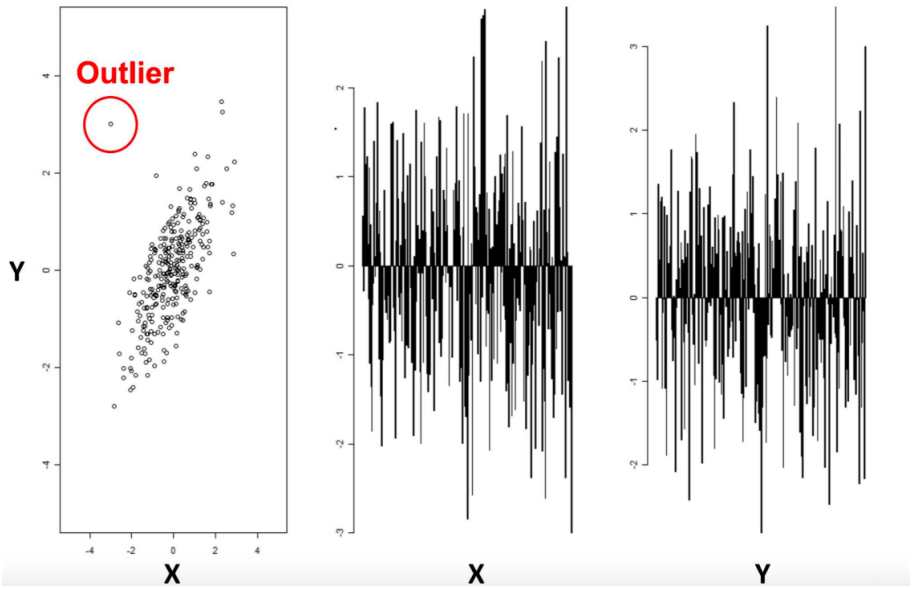


Figure 2.5: Anomaly detection independent of time vs time series
Source: https://towardsdatascience.com

The crash prediction using the provided data which is time-dependent data in which the crashes are very rare is closely related to a gear bearing failure [18] problem in a NASA machine. To develop the relation in these two data sets, the problem of bearing failure needs to be understood. In this example, the vibration data of bearings is provided for the lifetime of the bearings. A bearing failure is a very rare event occurring after many years. A lifespan of a single bearing is shown in the figure below.

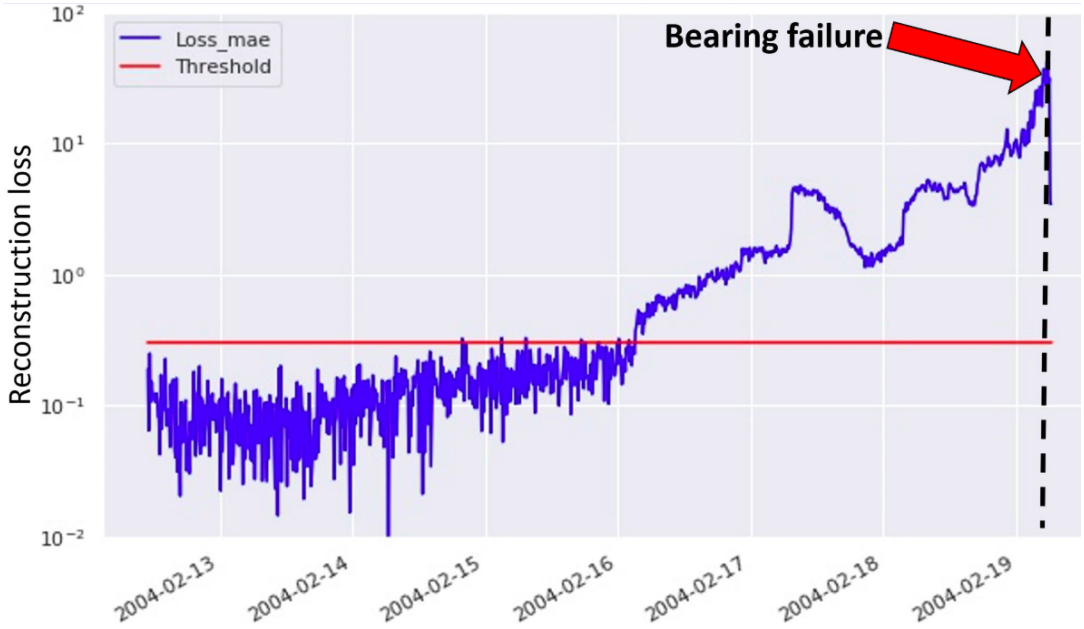


Figure 2.6: Bearing failure over the lifetime vs reconstruction loss
Source: https://towardsdatascience.com

The case of bearing failure is the motivation to adopt the anomaly detection technique used in this thesis. The free data set provided by NASA helped to select the algorithm. Using the NASA data as the base to perform experiments and learning the algorithm behavior by feeding this data at the input.
The algorithm selected for the motorcycle crash prediction is Convolutional Autoencoder (CAE) [19]. The reason for using this algorithm is that its performance for multi-dimensional data is much better in predicting anomalies from the normal data. Another reason to prefer this algorithm on classification algorithms like one-class SVM is that it minimizes the training time by using all the resources and has the ability to perform parallel processing. Also, it deals with non-linearity and complex data sets.

2.5 Convolution

A set of learnable filters forms the convolution layer [20] in the CAE algorithm. The convolution filters can be of different heights, widths, and depths. The height and width of the filters are independent of the size of the input data. However, the depth of the filters is the same as the input fed to the filters. Commonly the height and width are smaller whereas the depth is the same size as compared to the input data. The commonly used filter sizes are $3 \times 3 \times d, 5 \times 5 \times d$, or $7 \times 7 \times d$, where the three positions correspond to height , width and depth (d), respectively. During the process, the filters are convolved over the height and width of the input signal. Convolution takes place by taking the dot product of the filter with input at every layer. The result is called a "feature map". These feature maps stack up to form the final output. The values of these feature maps are computed by using the following formula. Where the input signal is denoted by f and our kernel by h. The indexes

of rows and columns of the result matrix are marked with m and n respectively.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

The figure shows the mapping of input to the reduced dimensions passing through the convolution layer to the final output. Here only a local region of the input signal is mapped to the hidden neurons. This technique is inspired by computer vision which is less complex than connecting fully connected layers and each neuron to every input feature. Moreover, the later discussed is extremely costly in terms of complexity and requires more processing power to compute.

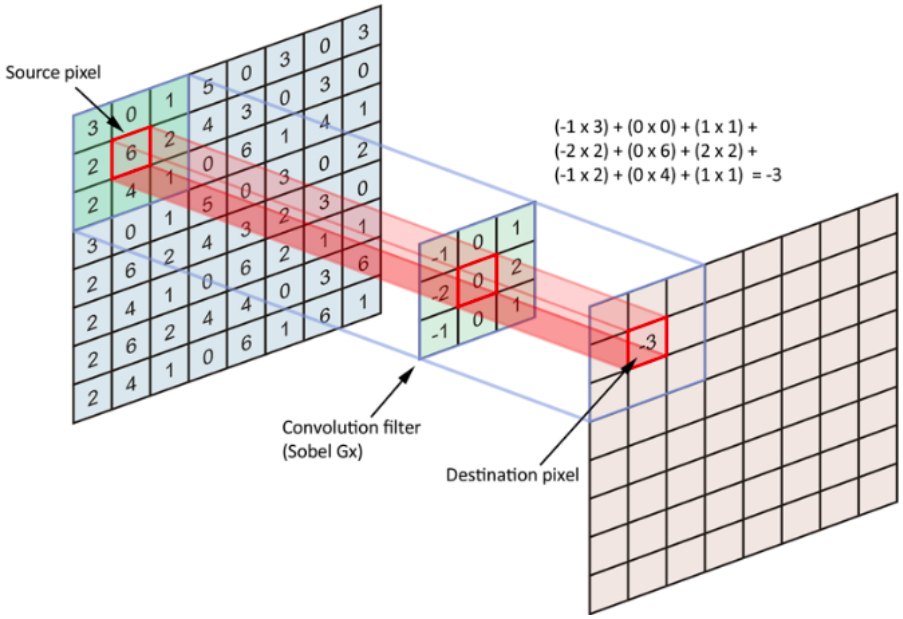


Figure 2.7: Convolution operation

The encoding layer used in this project has two 2D-convolution layers. In each layer, 8 times convolution is applied with the filters.

2.6 Autoencoder

An autoencoder is a type of artificial neural network which is used for dimension reduction [19]. It works by learning efficient data coding in an unsupervised manner [21]. There are two main parts of CAE which are called layers and the one attached to the input is the encoding layer. The purpose of the encoding layer is to learn the reduced parameters or dimensions of data which will be used to create the same data at the output of the algorithm as it is on the input. The layer at the end of the CAE is the decoding layer which is attached to the output of the algorithm. This layer uses the reduced parameters learned by the encoder to create similar data at the output of the CAE, as it was fed at the input.
In this example of bearing failure, the CAE artificial neural network compresses the sensor readings which is basically bearing vibration data to a lower dimension

representation, It develops the correlation among different variables at the input and saves their relation in lower dimensions that are enough to generate the same results at the output of the CAE. The main difference between the principal component analysis (PCA) model [22] and CAE neural network is that it also allows the input variable's non-linearities.

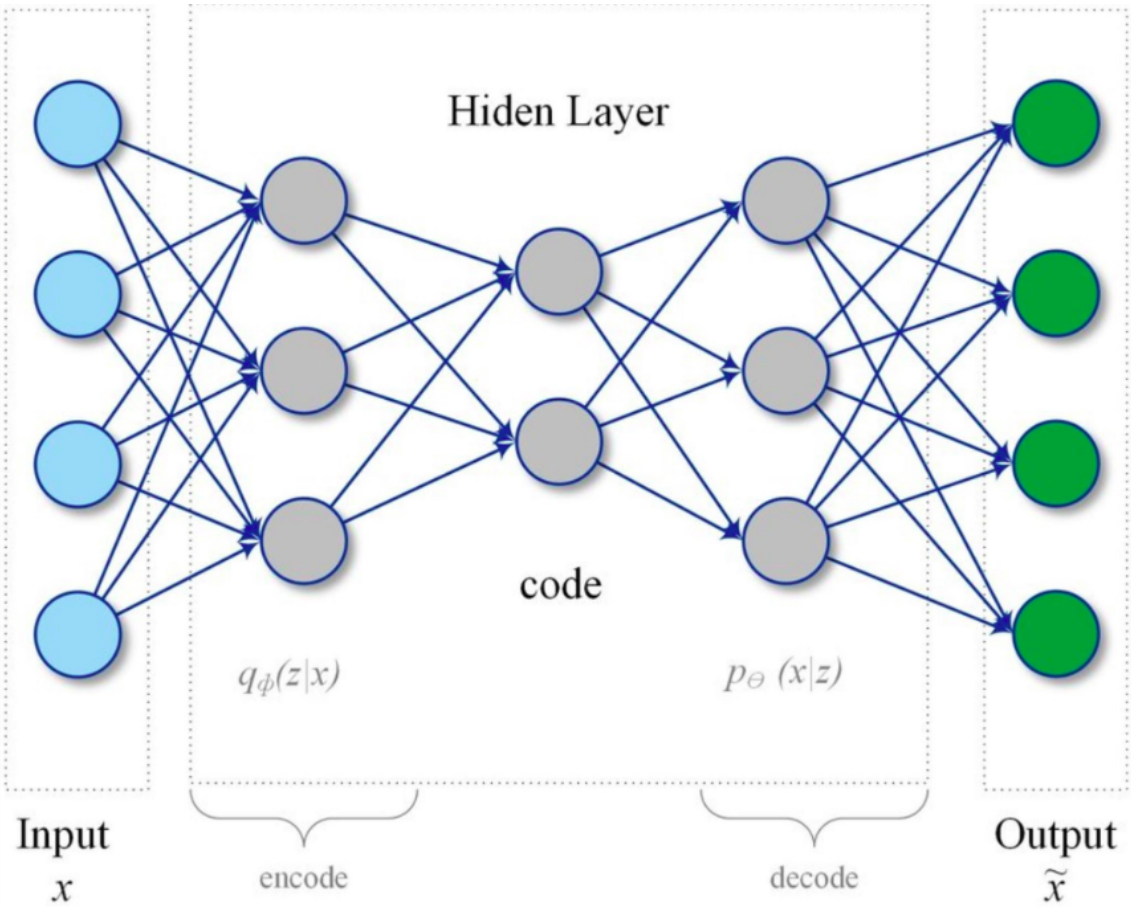


Figure 2.8: Autoencoder network
Source: https://towardsdatascience.com

From the figure 2.8, it can be seen that the input is connected to output using a hidden layer, in other words, it can be taken as a black box that connects the inputs to outputs. This black box learns the representation of the data in terms of its parameters as its dimensions. It works by reduction techniques, learning the reduction of dimensions, and reconstruction of the data at the output using these learned reduced dimensions. Autoencoder can be single layer perceptrons (SLP) or multilayer perceptrons (MLP) [23] depending on the layers in that black box. Since there is one layer each for input and output, controlling the number of hidden layers can make it a multi-layer or single layer. In most cases increasing the number of layers can increase the reduction of dimensions. In terms of dimensions, input and output layers have the same number of nodes which are known as the dimensions in each layer. Although an autoencoder aims to generate as close as possible results at the output, as of input, but still there is a margin of error.

2.6.1 Mean Error Distribution

The evaluation of the results of the autoencoder depends upon the difference in the shape and characteristics of the data generated at the output using the data at the input layer. The data generated at the output layer is automatically normalized by the autoencoder, therefore the input data or the training data is also normalized to compare it with the output. The mean error distribution (MED) of this difference from output to input is calculated. The range of this MED is used to evaluate the results. If the class lies within the error distribution range it is predicted as normal data and if it is outside the range of MED, it is predicted as anomaly data. Here the error from output to input is not as important as the range itself is concerned to evaluate the performance of the algorithm in detecting the anomalies.

3

Methods

The process of crash prediction begins with the processing of the data provided by Detecht, making it useful for the objective. The objective is to make data compatible to be fed as the input to the autoencoder. Implementation of the algorithm, tuning the algorithm parameters, and defining an optimum CAE model. Deriving the results from the autoencoder output and comparing the results to find the error, that is used as the evaluation parameter for the algorithm. To organize the data to be used as the training data or making it a suitable input for the CAE. The Detecht's data passes through a process. This process includes the following steps.

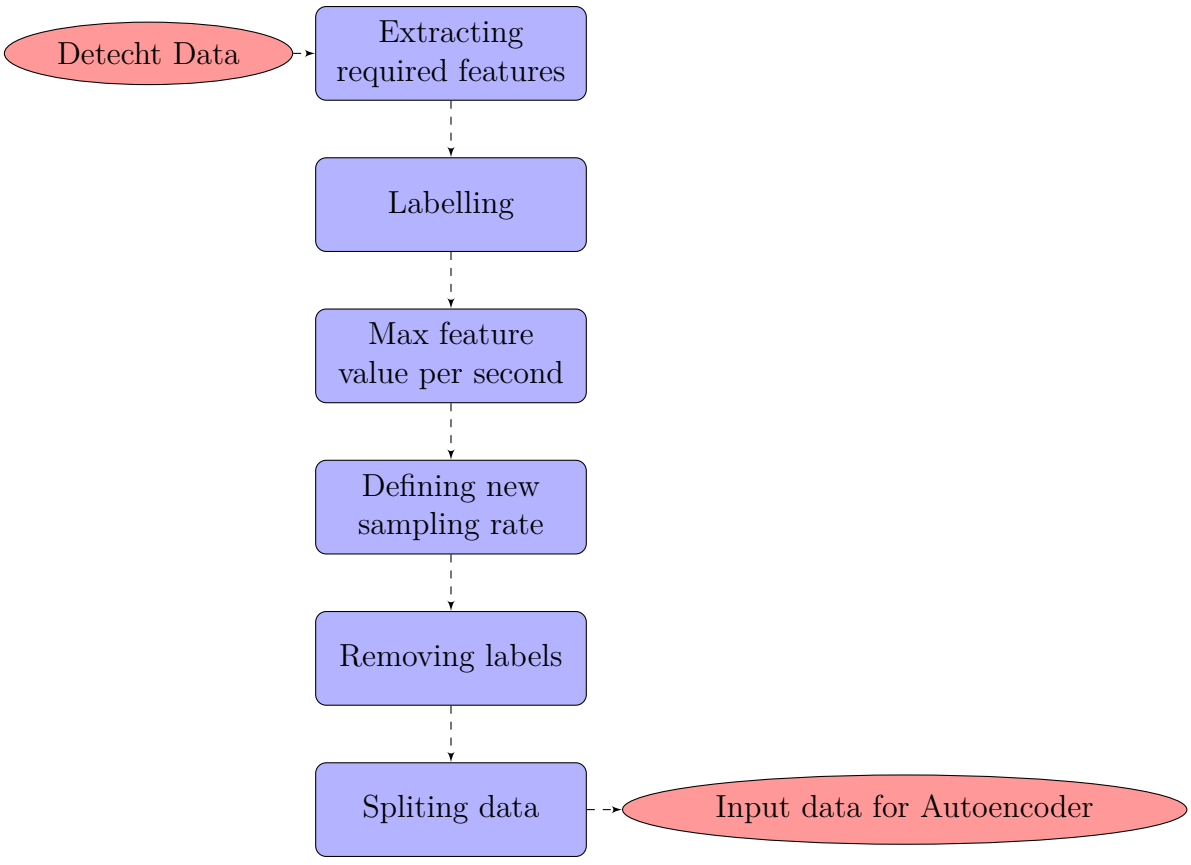


Figure 3.1: Flowchart of Data processing

3.1 Data

The data set for this project is provided by Detecht company. This data includes different features collected from the smartphones of motorcycle users who are using the Detecht smartphone app. This data is mainly from the users from Sweden, however, there are few examples from the riders using the detect app in Denmark as well. Other than normal data, Detecht has collected the data from test crashes simulated using a crash dummy. Detecht has provided their kind permission to use the data for this thesis.

3.2 Features

The data collected from smartphones provide information about different features or parameters. The app is designed for both Android and iOS smartphones. The features collected from the smartphones are values from the accelerometer sensor in the X,Y,Z directions, gyro sensor values in X,Y,Z axis, and GPS coordinates. The GPS coordinates are used to calculate the speed of the motorcycle rider. These GPS coordinates can be used to send the location of the motorcycle crash to the emergency contact number along with the rider's details. The absolute sensor values are also used to enhance the negative effect of the crash in the sensor values. These parameters are collected with time and the sampling rate varies according to the different conditions.

3.3 Labeling

The data is labeled into two classes, crash or no crash. Crash is considered as an event recorded in time having information of data for several seconds and not just an incidence or jerk of a second. This will get more clear in the following section selecting a time window (3.3.4). The crash class includes all reasons for accidents identified in the introduction section. It includes crashes at the junctions, failure to control over the bends, overtaking, high speed, bad weather conditions, and drunk driving, etc. The real crash examples are very few as compared to the normal class. Some of the test crashes are performed to collect more data from simulating dummy crash experiments. Having 26 simulated crashes and 3 real crashes. This makes it an anomaly detection case because of very few examples of crash class as compared to the non-crash class which has 940214 samples.

3.3.1 Sampling

The Detecht smartphone app is designed to collect data at a variable sampling rate depending on different conditions. Some thresholds are defined for accelerometer, gyro sensor, and speed, such that if the values exceed these thresholds it is more likely to have an accident. Keeping these thresholds in focus, the sampling frequency of the data increases automatically in such cases. This is to capture all the peaks and important samples at the time of the crush. Moreover different smartphones have the

capacity to process data at different rates and the values from the sensors of different smartphones are also not the same for the same event. The sampling rate using the Detecht app also varies in different versions of the data. Another problem is that different smartphones generate data at different frequencies. Keeping all these above variations in sampling rate in focus, the data needs to be processed to behave as regular time-varying data. The difference in operating ranges of some smartphones is identified by Detecht, depending upon the data collected from the sensors of those smartphones. The data collected from Detecht's application installed on different models of iPhones including iPhone 6, 6 Plus,7 and 8 provided different sampling rates at the time of the crash.
Since the sampling rate varies for smartphones of different brands and the Detecht application is designed to collect more samples per second during the interval when there is an irregular change in sensor values. This creates much more samples for the algorithm to process in the cases when the feature values exceed the threshold. The solution to this problem is to make the sampling rate uniform. For this, the approach that is adopted in this thesis is to take only the maximum of each feature value in all X,Y,Z directions in every second. This keeps only the sample having maximum values in every second and discard other samples from the same second. By doing this the important information in the peaks during a crash event are saved and unnecessary samples are deleted to reduce data.

3.3.2 Removing Labels

It is no use to train the algorithm on the data which is already labeled with crashes and no crashes, therefore the labels from the labeled events are removed. Data is divided into training and test data. The real crash events are separated manually to be fed with test data. Apart from real crashes, the dummy crashes are also processed and saved with the real crashes.

3.3.3 Seleccting A Time Window

A crash event is not just a sudden change in features. In other words, not every jerk or sharp peaks of accelerometer and gyro-sensor values indicates a crash. To make it confirmed as a crash event and reduce the false alarms, it is important to study a few seconds (s) after the sharp peaks as well. A time window of the 30s is selected after every second and these 30s windows are saved as new samples or events to be fed to the algorithm as input. The purpose of selecting this window of time is to cover two parts of the crash event in them. The first part is the crash jerk which is indicated by sharp peaks in the sensor outputs. The second part is the few seconds after the peaks where there is no significant change in the feature values. This no-change indicates that the rider has not started driving the bike again normally after the crash. This no change just after the sharp peaks confirms that the crash has occurred and the rider needs to be rescued.
After examining the simulated crash data, it is decided to break data into 30 seconds overlapping windows taken after every second. Removing all null data and useless data (When app is recording data while rider is not driving at all) to improve training

time and quality of the data. This provided the 940214 samples used as training data for the algorithm.

3.4 One Class SVM

The process of decision making to use which of the machine learning algorithms for this crash prediction anomaly detection problem, involves the training time and the performance of the algorithm to learn data in one class as normal class. This processed data having samples of 30 seconds is used for training using one class SVM. The algorithm failed to learn the data completely. Unsuccessful training using one class SVM lead to use of much complex anomly detection algorithm for this problem which is convolutional autoencoder.

3.5 Flowchart of Convolutional Autoencoder

The data is now processed and is in the shape to be used with the algorithm. This data is fed to the input of the CAE. The CAE follows some steps of operations creating filters and layers. These steps include the input data connected to the encoder, from encoder layers to the black box known as code, from the code layer to the decoder, and finally to the output. The flowchart below shows the sequence with these steps.

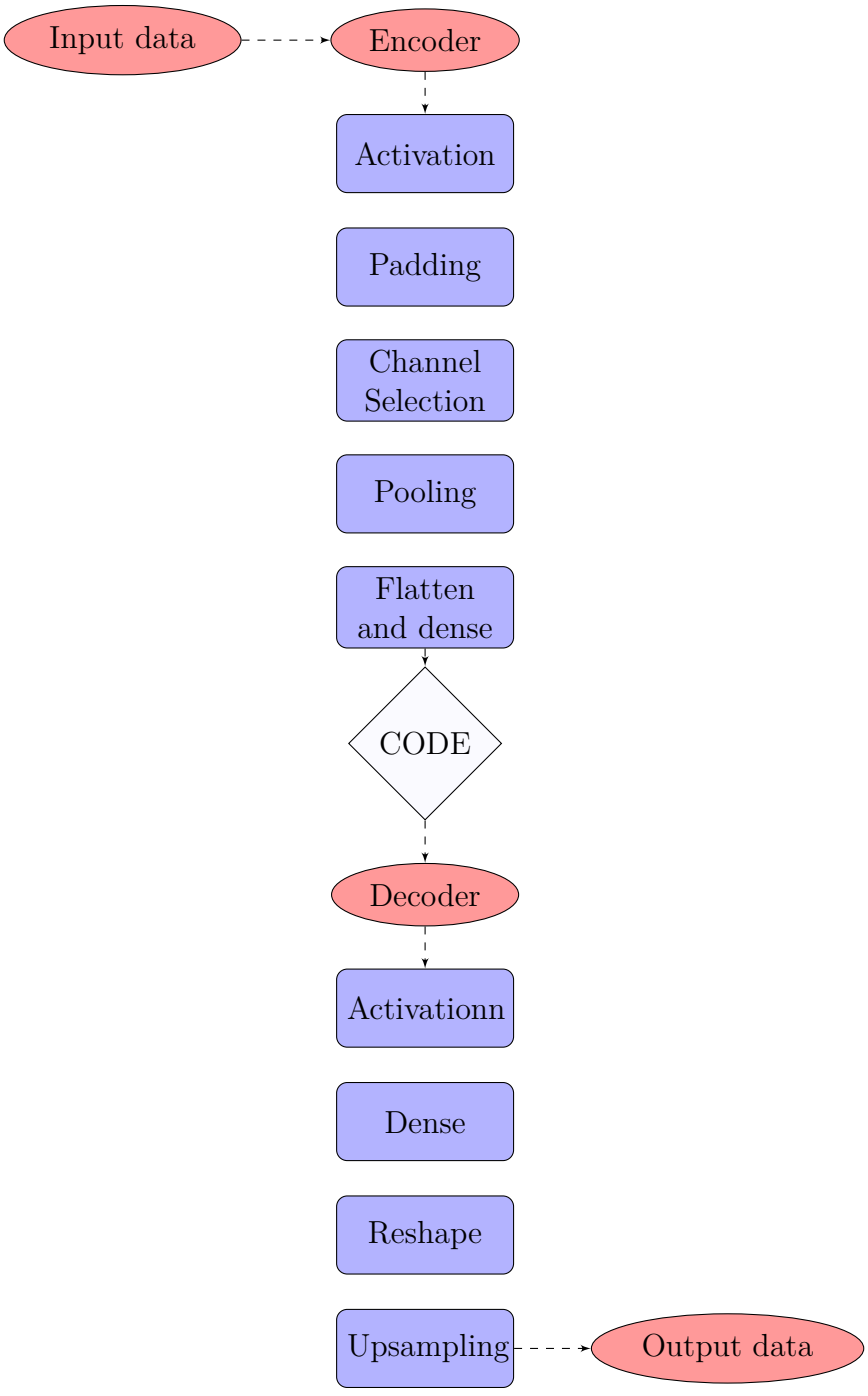


Figure 3.2: Flowchart of working of Autoencoder algorithm

3.6 Encoder

The data fed to the CAE is practically the input to the first layer of the encoder. This input data has a size of (1,30,6) in this case. In this size 30 represents a window of 30s and 6 is the number of parameters of data. In the Encoder layer, some adjustments are done that controls the layers, the sizes of the layers, and nodes in each layer. The CAE used in the thesis is 2D convolutional autoencoder [24]. The input to the encoder has a size (8,3,3) where (3,3) is the length and width of the filters. Here 8 is the depth showing 8 filters will be learned in each layer depending upon the performance.

3.6.1 Activation

If the neural network is not activated it behaves just as a linear function. This means that the high dimension complex data can not be handled with such a neural network. The activation function [25] used for this purpose is usually non-linear function. These functions provide or enable the abilities of the CAE to handle high dimensional complex data. The activation function used for the activation of this CAE for crash prediction is rectified linear unit (ReLU) [26]. This is a non-linear function and it allows only non-negative values at the output. The ReLU function is shown in figure 3.3 below.

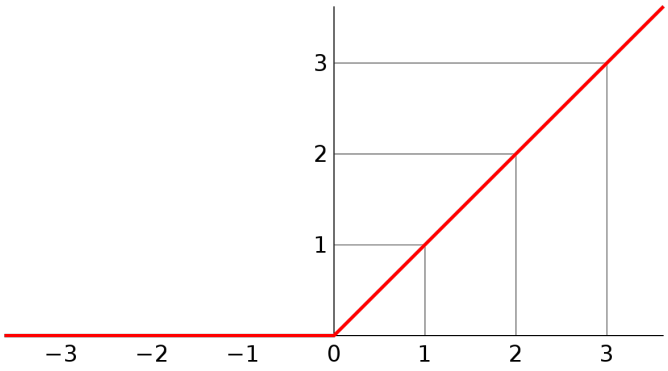


Figure 3.3: ReLU function

3.6.2 Padding

After the convolution operation is applied, the dimensions of the input data are reduced. These reduced dimensions are learned from the filters during passing through each layer of filters. Each filter reduces the dimensions, therefore it needs the data to be padded to keep it in a uniform shape. For this, the padding [27] setting is set to "same" which pads the reduced data with zeros at the ends shown in figure 3.4.

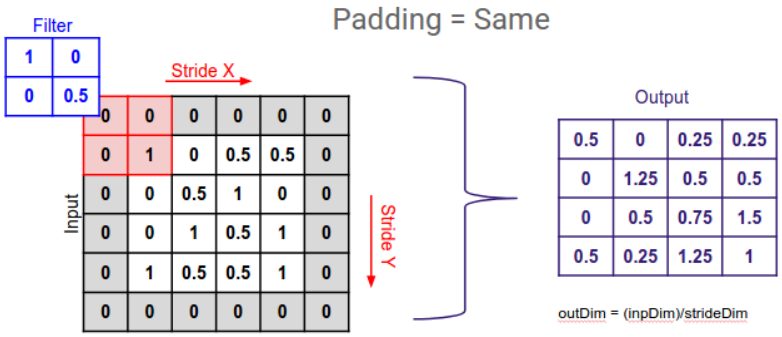


Figure 3.4: Padding with settings set to "Same"
Source: https://towardsdatascience.com

3.6.3 Data Format

Here the channels [28] are selected for the operation. For example for RBG colored image as an input data, the channel is set to 3. For grey mode, the channel is set to 1, which is set to 1 in this case as the input data is not an image.

3.6.4 Pooling

The approach to reducing the size of the feature map, reducing parameters in the model is known as pooling [29]. Pooling impacts the complexity and fitting of inputs of the layers of CAE. There are three types of pooling maximum, minimum, and average-pooling. Out of these techniques, max-pooling is used to keep the number of parameters of the network controlled. Max-pooling is also used to get maximum performance as compared to the other two pooling preferences. The depth of the output of pooling is the same as the input because it is applied independently to each layer. Using max-pooling with 2x2 filters and stride equal to 2 picks the maximum values of features from a non-overlapping 2x2 box and saves it. This is demonstrated in figure 3.5 by using different colors. For example in the first pink colored box maximum value is 6 which is then copied and saved as a reduction of this pink box. Similarly, 8 is saved as the reduction from the green box and so on.

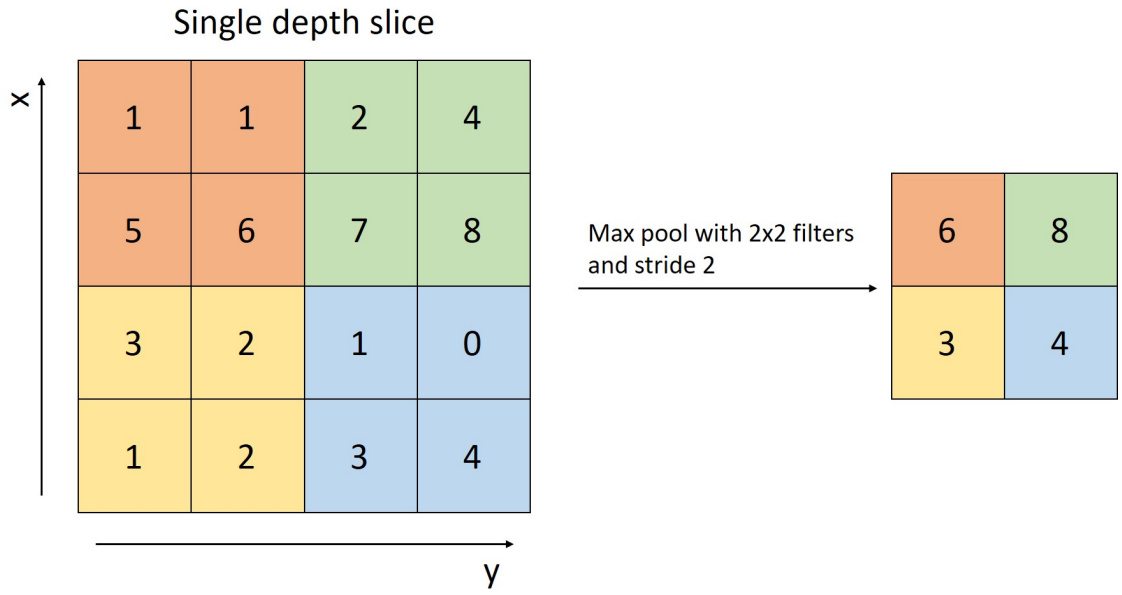


Figure 3.5: Maximum pooling using 2*2 filters

3.6.5 Flatten and Dense

The final layer of the Encoder flattens the 2D convolution output to the 1D output. In the dense layer, a fully connected neural network is used, This takes the dot product of 1D output with the set neurons. The set neurons or weights in this layer are 256.
This gives the result from learned reduction and stores the results in "Code". Code is the middle layer that stores the reduced parameters and the information required to regenerate the data from these reduced parameters.

3.7 Decoder

After passing through filters of the encoder, the input data is compressed which means the reduced dimensions are learned by the autoencoder. These reduced dimensions are used by the decoder layers connected to the output to generate similar data as of input again at the output. Like the encoder, the decoder also has different steps that it follows in the process of generating the data again.

3.7.1 Activation

The decoder is the same as the encoder but the operation order of the steps is in reverse. The activation function for the decoder is also the same as the encoder which is ReLU. The reason for adopting this function is that it decreases the training time. However, the activation function for the last layer of the decoder is sigmoid as we need to determine the probability in this layer as probability lies in the range [0,1]. This activation function of sigmoid has output values in the range [0, 1] which

fits perfectly with the probability range. The sigmoid function is shown in the figure below.

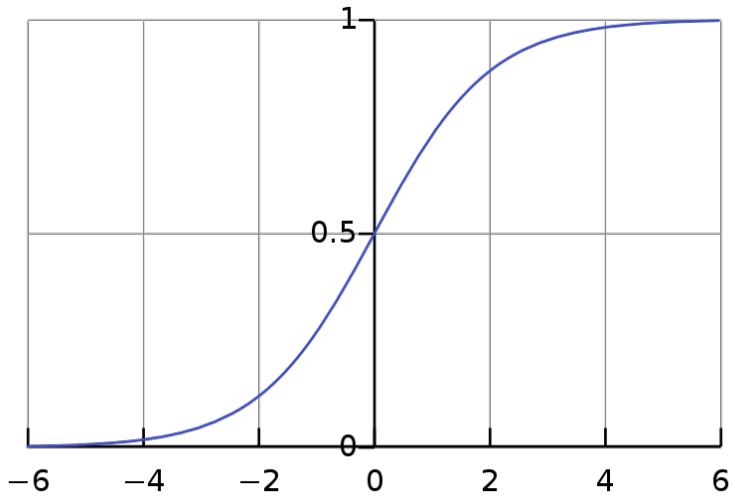


Figure 3.6: Sigmoid Activation Function

However, the activation function used for the decoder layers is later changed to ReLU which is the same as used in encoder layers. The reason for adopting this function is that it decreases the training time.

3.7.2 Dense

The dense layer at the decoder works in the same principle as of encoder. The output from the 1D fully connected neural network is flattened and the filters in the layer take the dot product with the set weights. The output is generated from the process in 2D. The input of 1x30x6 where 30 represents seconds in each sample and 6 represents features is now multiplied with weight 45. This weight is formed by 1x15x3 which is the reduced half size of the input.

3.7.3 Up-sampling

This stage of the decoder starts regenerating the shape of the actual input data. This is done by using a set of filters in this layer. The up-sampling [30] process consists of convolutions having a different number of filters. Each time the data is multiplied and filtered to form the output. Up-sampling is an important part of the decoder that helps to generate the output using the learned features, as the shape of the data was reduced using max-pooling in the encoder.

3.8 Optimized Model

The figure 3.7 below shows how the pooling is performed after every convolution. These layers and filters are arranged to form the encoder and decoder of the CAE.

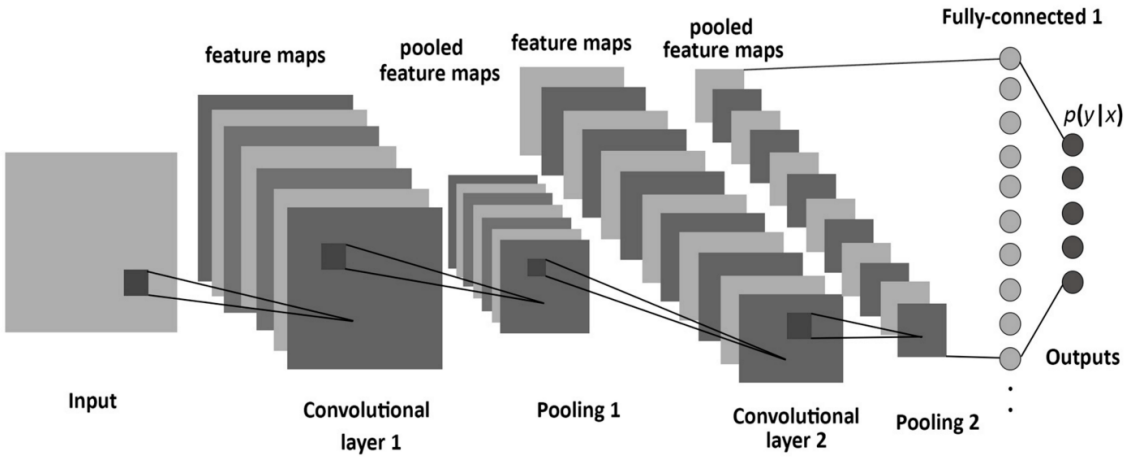


Figure 3.7: Encoder Layers from Convolution to fully connected network. Source: https://towardsdatascience.com

Using the above set of layers applying convolution in encoder and decoder produced a large error, which affects results and makes it difficult to identify an outlier. Therefore the model used for the project consisted of two convolutions of 8 filters each followed by max-pooling. This process of convolution and max-pooling is repeated 2 times in the final optimized model because the error was reducing after each training. However, no significant change in error is observed after applying more than 2 sets of these operations (each set is 2 times convolution followed by max-pooling). Therefore its avoided to set more layers, which add the extra learning time for the algorithm. These set of convolution layers are represented by red blocks followed by max-pooling layers shown by green blocks in figure 3.8 below.

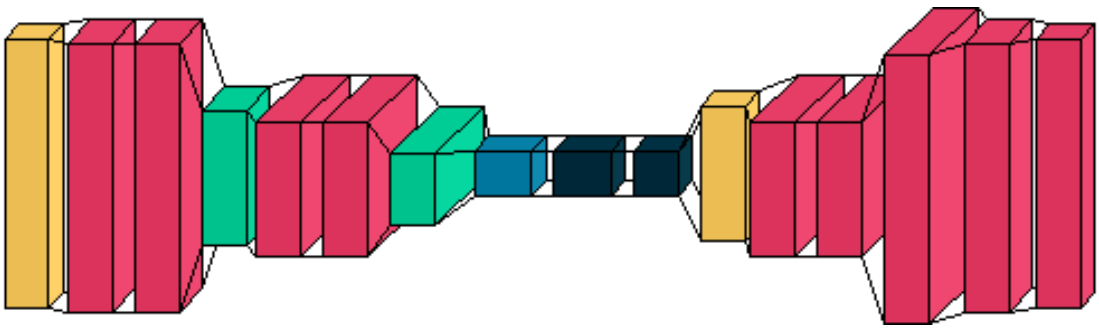


Figure 3.8: Optimized implemented model

The decoder of the optimized model consisted of a similar pattern as of encoder. The difference is that instead of the max-pooling, the decoder uses up-sampling. So the three sets are repeated forming 2 convolutions of 8 filters each followed by an up-sampling layer. All the layers of convolutions, pooling, and up-sampling are two-dimensional except the fully connected network used in the encoder.

3.8.1 Multicore Usage

The algorithm is tuned to train parallel [31] on all cores of the CPU, this has reduced the processing time, otherwise, it would take much more time for training whenever any setting was changed. The code is running in tenser-flow [32] to use GPU for processing. Tensor-flow is an open-source platform for machine learning algorithms that enables the use of GPU for faster processing and provides flexible tools for working with multi-dimensional data.

3.9 Evaluation

To evaluate how accurate reconstruction of data is obtained at the output, the output needs to be compared with the original data. As the output data from the CAE is normalized, therefore the input data is also normalized to compare it with the algorithm output. The mean and standard deviation (STD) of the input data is computed and then it is normalized using the formula:

$$Normalized\,data = (Data - Mean)/Standard\,deviation \tag{3.1}$$

3.9.1 Error Distribution

The evaluation of the results to detect algorithm performance in predicting a crash consists of calculating the reconstruction loss [33]. This process of calculating the error distribution is for all data points in the test data set including 26 simulated and 3 real crashes and comparing this error range or loss to the previous algorithms used by the Detecht company for flagging this as an anomaly.
The normalized input data or the Detecht's data is compared with the output data which is the result of reconstruction from the autoencoder. This comparison gives the difference between the newly regenerated data from the algorithm and the actual data. Mean Error distribution (MED) is the parameter used in this crash prediction algorithm to evaluate the performance of the CAE used. The MED provides the range of error produced in the reconstruction. The outputs from the test data are compared with this range of error distribution and the events having error exceeding this range are declared as an anomaly or crashes.

4

Results and Discussion

The normalized input data or the Detecht's data is compared with the output data which is the result of reconstruction from the autoencoder. This comparison gives the difference between the newly regenerated data from the algorithm and the actual data. Mean Error distribution (MED) is the parameter used in this crash prediction algorithm to evaluate the performance of the CAE used. The MED provides the range of error produced in the reconstruction. The outputs from the test data are compared with this range of error distribution and the events having error exceeding this range are declared as an anomaly or crashes.

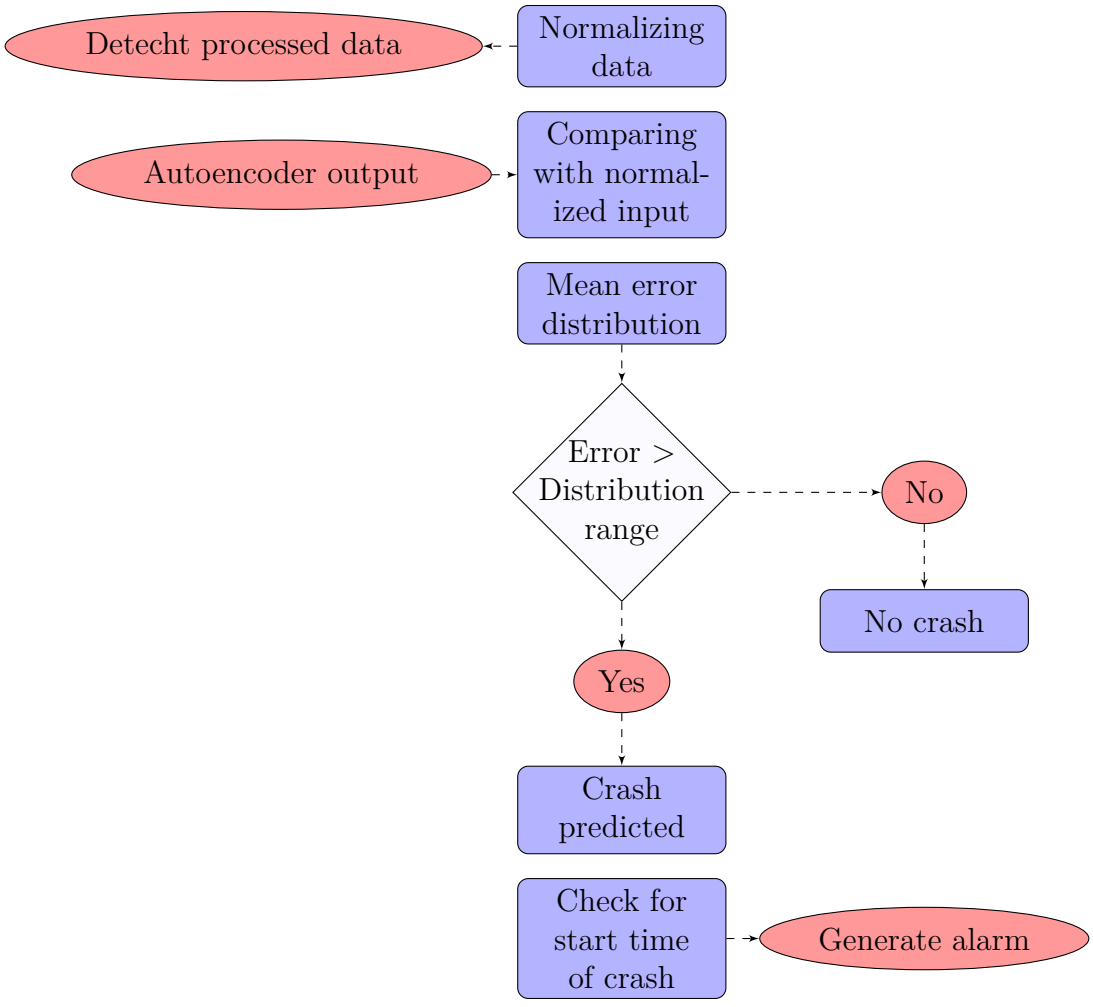


Figure 4.1: Flowchart of predicting crash.

4.1 Error Distribution

The error distribution or the reconstruction loss [34] of crash data makes it possible to define a threshold value for an event to be declared as a crash event or normal driving. From the distribution plot below in figure 4.2, the error range on the x-axis which is from 0 to 0.2 is important to study. This range defines the threshold for an event to be an anomaly or not.

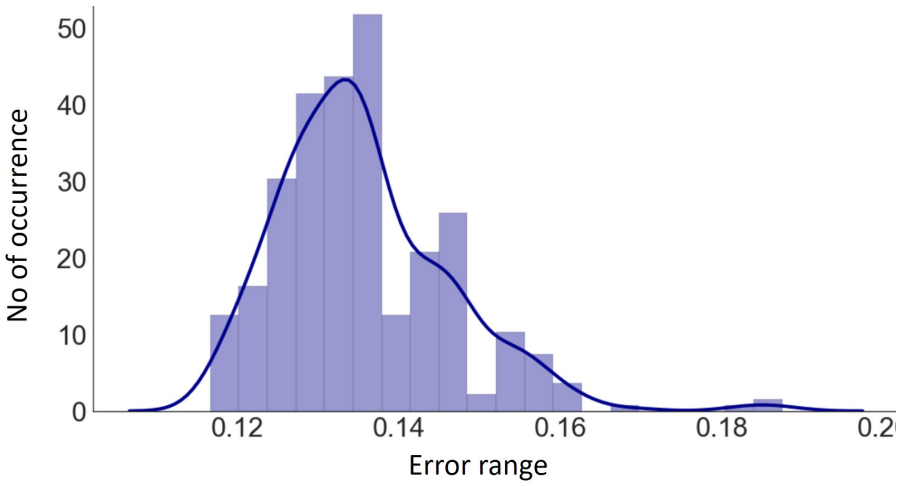


Figure 4.2: Mean Error Distribution

In the above figure, the blue bars correspond to the number of samples from the training data that occurred for each value of the error range. Since the training data has no crash events in it as it trained with only one class, therefore every value outside this range is considered as an anomaly. The blue line represents the defined threshold value or range for flagging an anomaly. The crash event is any event that lies outside this 0-0.2 range of mean error distribution. The peak magnitude of occurrence is 50, but it is of no importance for predicting a crash since it only shows that this much samples of the training data belong to the corresponding value. These values for the number of occurrences are proportional since actual figures are in thousands depending upon the number of samples in training data. The range is very important to predict the class as a crash. The table 4.1 shows the exact values of error calculations from the results.

Error	Value
Mean error	0.208
Maximum error	9.826
Minimum error	0.008
Standard deviation	0.405
Median	0.081

Table 4.1: Table of error values

4.2 Anomaly class

Predicting crashes is the aim of the thesis and the crashes being very rare and few in numbers are the anomaly in this case. To study the behavior of the crashes, Detecht provided 26 test experiments which are confirmed crashes. These test examples are tested with the algorithm and most of them are predicted as crashes as the loss distribution of these events is outside the 0 - 0.2 range. The loss distribution plot of one crash event is shown below which defines that the event is predicted as a crash event by the autoencoder.

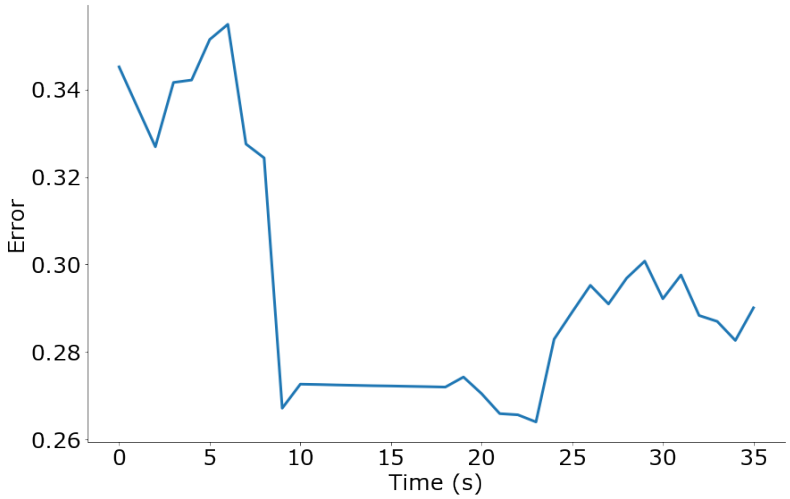


Figure 4.3: Error distribution of a test crash

In the figure, it can be seen that the maximum error magnitude is at 6 seconds shown on the x-axis, and the error value at the y-axis is approximately 0.35 which corresponds to this time. This maximum error of 0.35 is outside the distribution range which is 0- 0.2 therefore the algorithm has predicted it as a crash event. However, in the above plot and some other crash events, it is observed that the magnitude of error remains outside the error range even after the crash. There are several reasons for this behavior when the start time of the crash is not confirmed. Although this is considered the first step in crash prediction. To confirm it as a crash and avoid false alarms from the algorithm some procedures to determine the start of the crash are followed in the later steps.

4.3 Comparison

The results that are confirmed as crash events by the CAE are compared side to side with one of the previously proposed algorithms by Detecht. This comparison is performed on all of the test crashes that are collected by experiments. The purpose is to study the behavior of the CAE algorithm response to the crash event as compared to the values from the accelerometer and gyro sensors at the time of the crash. An example from the comparison is stated below by relating figure 4.4 and figure **??**.

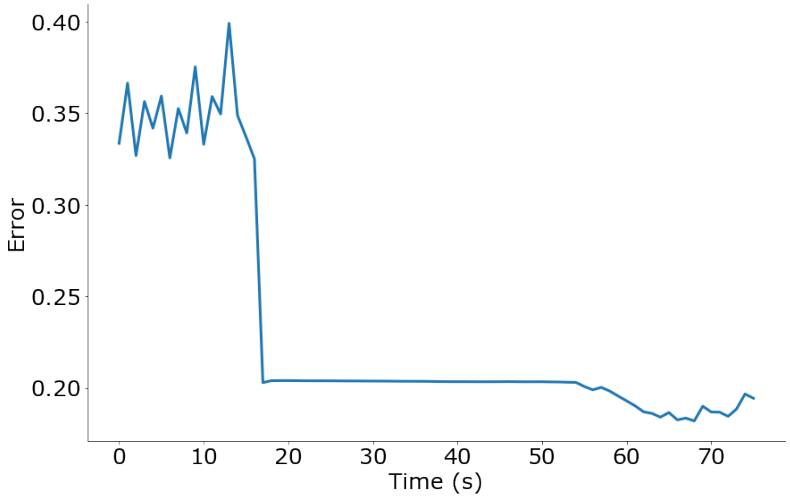


Figure 4.4: Error distribution of a crash event

The maximum error in the plot of figure 4.4 is at 17 seconds which indicates that the crash happens at this time. However, the error started rising from the error range of 0.2 even from the start of the plot. The reason behind this rise is that every second contains the information of the next 30 seconds since we have taken each sample for 30 seconds. According to this, the crash will be at maximum or the end of the rising pattern of the curve but it starts affecting the error 30 seconds before it actually happens. This is also manually confirmed by comparing it with the sensor values from the data.
In this example, in figure 4.4 the error remained below the 0.2 range after the crash but this is not always the case. As in most of the experiment crashes or some real crashes where the rider is not seriously injured, the rider starts riding again after a small pause and the error value gets disturbed. This might be because the rider has no major injuries and does not wait for the rescue. The algorithm worked for such cases as well.

4.4 Start time of the crash

The input to the algorithm are events, each of these events is 30 seconds long and are taken after every second interval. This means that the algorithm has the information of the next 30 seconds when it is ready to predict any crash. The crash is indicated by the peak in the accelerometer and gyro sensor values. These peaks are translated 30 seconds before the actual time in the error distribution graph as every second has information of the next half minute. This means that the algorithm requires information of 30 seconds to report a crash. This method adds a 30 seconds delay in the process to call rescue during live ride, but it has removed most of the false alarms generating when there is no crash. The figure 4.5 below indicates a sharp peak at 36 seconds exceeding the distribution range indicating it has a crash at this point. But the algorithm has already given an error greater than the distribution range of 0.2 at 6 seconds. This indicates that the effect of a crash is translated 30 seconds before it happened and this error increases until it reaches the point where

the crash actually occurred. The error of the crash event is shown in figure 4.5 and after comparing it with the sensor values this behaviour is confirmed.

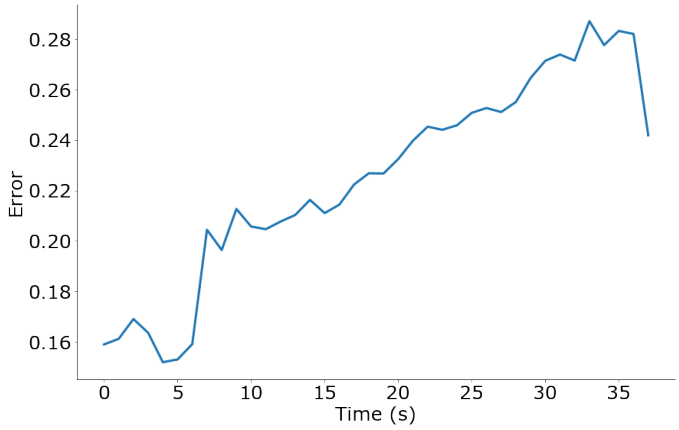


Figure 4.5: Start time of crash from CAE output

The figures 4.5 is just one example indicating that the crash occurred at some time and it has effect on the CAE output 30 seconds before it. However, the threshold values to trigger for the crash event or to indicate the start of the crash should be generalized for all experiments and real crashes. Therefore after observing the behavior of all of the experimental crashes, the generalized algorithm is designed for the prediction of the start time of the crash. This algorithm starts looking for the increasing pattern of the error starting from the point where its value gets higher than the error range of 0.2 and declares the maximum peak which is within the 30 seconds after this point, as the start time of the crash event. In simple words, this point is predicted as a crash and the algorithm generates an alarm in this case. The working of the start predicting algorithm is shown in the flowchart 4.6 below.

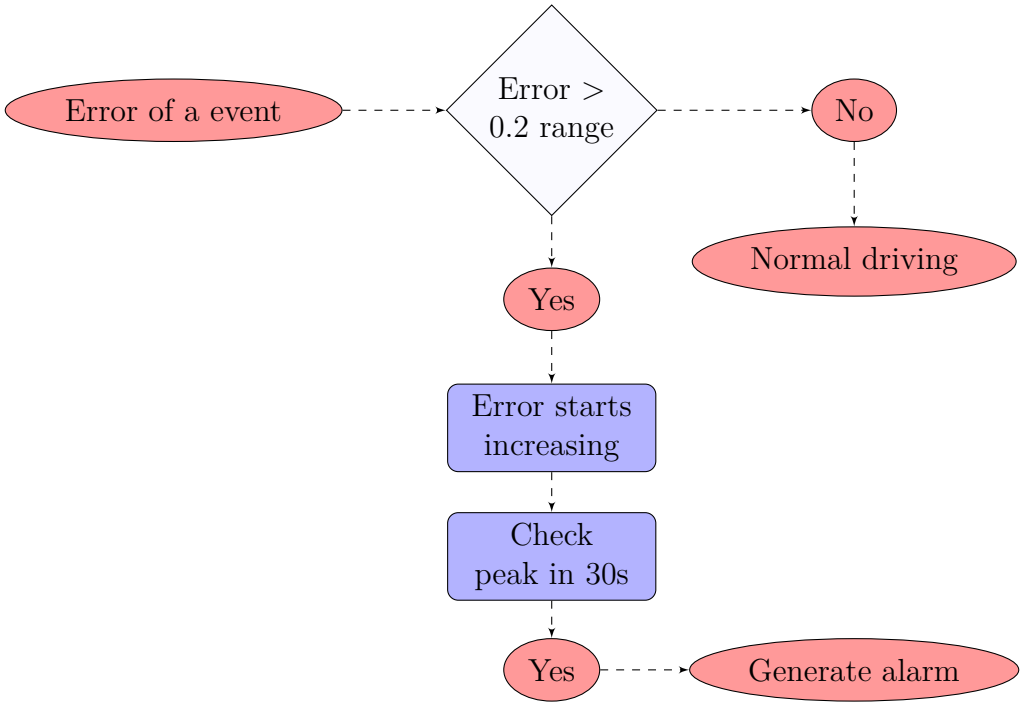


Figure 4.6: Generalized algorithm to predict start time

4.5 Limitations

However, there are some cases where the generalized method is not fully recommended for all of the test crashes. The reason for applying this method to identify the start of the crash and trigger the alert alarm is that it worked for most of the test cases without generating any false alarms. But there are few limitations of this method where the results are not satisfactory. Even using the optimized 30 seconds window which improved results but the sensitivity of only 62% is achieved for simulated crash predictions. Such cases are discussed below in order to understand the reasons behind the failure of the algorithm in these cases. These crashes are categorized as three types : crash at the start, crash at the end, and acceleration after crash. The percentages of these events predicted in the simulated data along with their respective sensitivity are mentioned in subsection "Summary of crashes" below. Apart from these simulated crashes, 135667 samples out of 940214 samples of the data were predicted outside the 0.2 error distribution range having sensitivity of 85%. These samples cause false alarms.

4.5.1 Summary of Crashes

The crash and no crash plots of the reconstruction loss are compared with the previously used algorithm by Detecht. The comparison involved the test experiment crashes carried out by Detecht experienced motorcycle rider using the same smartphone for all of these tests so that there is no problem in the calibration of sensor values or sampling rate variations by using different smartphones. The results are summarized as 62 % of the test crashes are predicted as true crashes having the

exact starting time of the crash. However, in the remaining crashes the algorithm has predicted these crash files as crashes but at the incorrect time where the crash not actually occurred. These false alarms are further divided into three categories as mentioned in the limitations section. The sensitivity of data in these categories is as below in table 4.2.

State of crash/ Reason of failure	Prediction	Sensitivity
Crash predictions	True positive	62 %
Crash at start	False Alarm	16 %
Crash at the end	False Alarm	8%
Acceleration after crash	False Alarm	14%

Table 4.2: Table of Percentage of data in different categories

4.5.2 Crash at the start

Practically the probability to have a crash in the first 30 seconds of starting the crash prediction smartphone application is very low.
But from the experiment crashes there are 4 cases where the crash is simulated in the first 30 seconds and the algorithm designed to check for a crash fails for these 4 crashes to be predicted correctly as a crash. Such a crash is shown in this figure 4.7. The reason is that the algorithm waits for 30 seconds to generate an alarm in order to reduce the rate of false alarms. If this wait time of 30 seconds to generate the alarm is reduced to 20 or 10 seconds then most of the normal driving behaviors where the rider accelerates much faster in the start, the algorithm starts predicting them as crashes. In this example, the algorithm fails to predict which of these peaks indicate a start time of the crash.

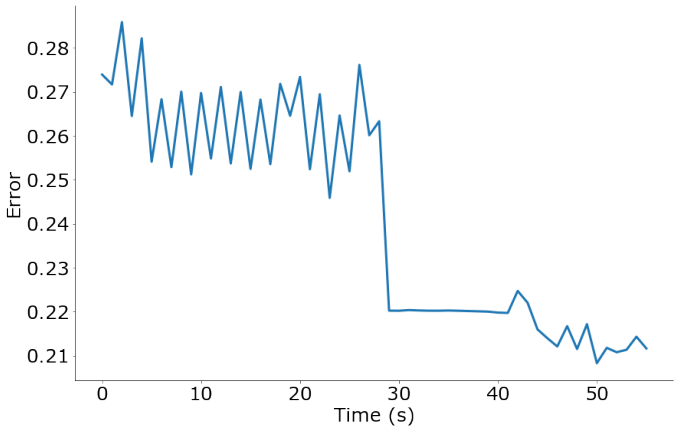


Figure 4.7: Crash at the start

4.5.3 Crash at the end

In 2 of the experimental crashes which occurred in the last 30 seconds of the riding or in the last 30 seconds of using the application, the behavior of the algorithm response is found abnormal.
There is not enough samples available for the algorithm to perform properly. Although it predicts the error higher than the normal driving indicating it as a crash event. However, the algorithm fails to predict the start of the crash because of the incomplete information fed to it after the peak of the crash. One such example is the error plot in figure 4.8. This shows an error peak at the end of the time but if the complete plot is observed, the error is higher than the 0.2 range even when the rider is driving normally. Another important observation from the plot is that, since each second has the information of the next 30 seconds therefore the last 30 seconds are never shown in the error plots. Even in this example, the file duration is 120 seconds but it only shows an error in the first 90 seconds.

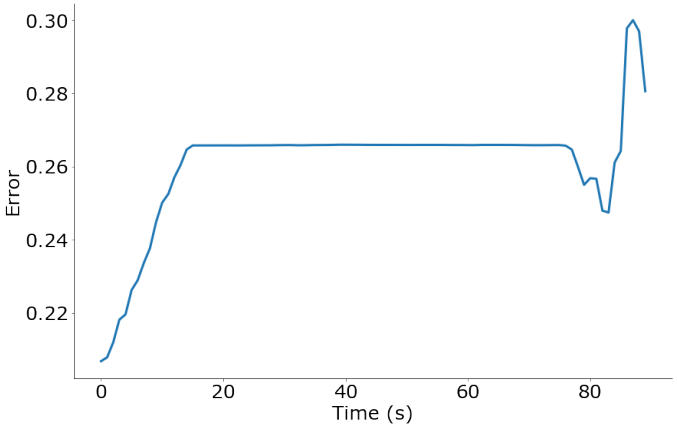


Figure 4.8: Crash at the end

4.5.4 Acceleration after crash

There are 4 test events out of 26, where the rider has a crash and after a little pause it starts riding again normally. Another possibility is when the rider starts with the higher acceleration and ends up in a crash because of any reason.
In this case, the algorithm fails to identify that which peak is a crash as in few events the normal driving style of the rider gives much sharper results in accelerometer and gyro sensor values than the values recorded at the crash incident. One of such cases is plotted in figure 4.9. In this file, the crash occurred at the start of the driving but after some rest, the rider starts driving again and the response from the algorithm is much higher for the normal driving after the crash. The possibility of this behavior of driving after the crash is very rare in real crashes. However, the experimental data provide such examples,

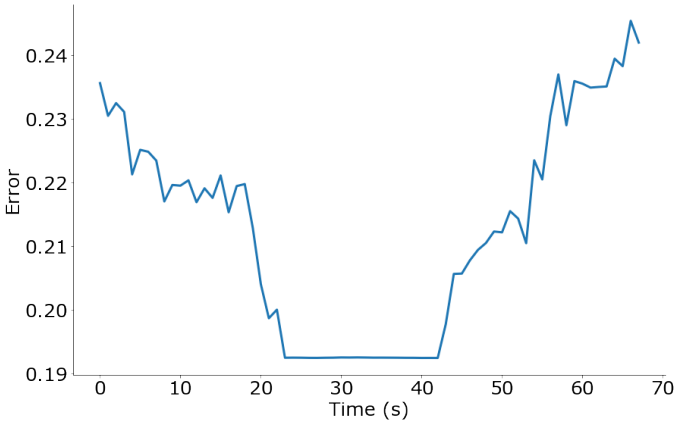


Figure 4.9: Acceleration after crash

and few of them generated false alarms as in this figure, the alarm is generated at the end of the time but actually, it occurred in the start.

4.5.5 Shrinking time window

At first, the algorithm is trained by taking samples of 30 seconds after every second. Later it is trained with samples of 20 seconds and 10 seconds respectively. Shrinking the window not only increases the training time to approximately two times as the number of samples is increased by a big difference, but also the reconstruction of the data at the output provides a high error range of approximately 0-9 which makes it difficult to draw a boundary between anomaly and normal data. Therefore the adopted 30 seconds window provides better results. The table 4.3 provides the error ranges by training the CAE with different settings before adopting the final model that consist of 30s window and 3 sets of (2 convolution layer with max-pooling) in encoder and 3 sets of (2 convolution layers with up-sampling) in the decoder of the CAE. This model is adopted as the optimum model for the crash prediction using CAE in this thesis.

Settings	Error Range
20s window	0-9
10s window	Noise (model learned no pattern)
30s window with 2 convolution layers	0-1.5
30s window with optimum model	0-0.2

Table 4.3: Table of error ranges by varing window size

4.6 Normal class

To test the algorithm, that is it only predicting the crash events outside the loss distribution range, or it can also predict no crash events. The processed data is fed to the algorithm and examined. A total of 804547/940214 samples are predicted correctly as normal driving data having no crashes making sensitivity of 85%. This data does not have any real or experiment crash events in it, as those are also removed at the start of the data processing.

5
Conclusion

Implementation of the Convolutional Autoencoder which is an artificial neural network was successful for the data provided by Detecht. The reason for using this algorithm is that it gives good results for multi-dimensional data and predicting anomalies. Minimizing the training time by using all the resources and have the ability to perform parallel processing. Implementing the algorithm and tuning it to have an optimal model based on the trade-off among training time, error range from reconstructing data, and the number of filters in each layer of the autoencoder. This also decides how many convolutions should be performed. The optimal model resulted in 85% of the data in the distribution range of 0 to 0.2 which is considered as normal driving. The remaining samples are out of the prediction range and declared crashes by the algorithm. This is because the data is not collected from experts but from real-world motorcyclists which can add human error while recording the data.
The performance of the algorithm is evaluated by comparing the results with the previously used algorithm by Detecht. The comparison not only provided the information about the performance but also the reasons for the failure of the algorithm in some test experiments to predict a crash at the crash time. The convolutional autoencoder has successfully predicted 62% of the total simulated crashes. The summary of predictions and failure reasons is prepared to have an understanding of evaluation.

Bibliography

[1] RoSPA, "RoSPA Road Safety Research," no. November, pp. 1–15, 2017.

[2] Westat and Dynamic Science, Inc., "Motorcycle Crash Causes and Outcomes: Pilot Study," no. June, p. 233, 2010.

[3] H. J. Hurt, J. Ouellet, and D. Thom, "Motorcycle Accident Cause Factors and Identification of Countermeasures: Volume 1 Technical Report," vol. January, no. Contract No. DOT HS-5-01160, p. 425 pgs, 1981.

[4] "Maids - In-Depth investigation of motorcycle accidents." [Online]. Available: http://www.maids-study.eu/index.php?error=hastolog. [Accessed: 06-Jan-2021].

[5] Li Jinhai, Wu Lintao, Wu Jingmei, and Mi Xiaoyi, "Study on the fatal crashes of rural roads in mountainous area," in Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 2011, pp. 1881–1885.

[6] A. Mubarak, N. Murali, and H. Anantharaman, "MCAS: A collision sustenance mechanism for motorcyclists," in 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2018, pp. 66–71.

[7] Li Jinhai, Wu Lintao, Wu Jingmei, and Mi Xiaoyi, "Study on the fatal crashes of rural roads in mountainous area," in Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 2011, pp. 1881–1885.

[8] G. Goy, C. Gezer, and V. C. Gungor, "Credit Card Fraud Detection with Machine Learning Methods," in 2019 4th International Conference on Computer Science and Engineering (UBMK), 2019, pp. 350–354.

[9] A. Nasser, D. Hamad, and C. Nasr, "Visualization Methods for Exploratory Data Analysis," in 2006 2nd International Conference on Information Communication Technologies, 2006, vol. 1, pp. 1379–1384.

[10] Haifeng sui, Bingru Yang, Yun Zhai, Wu Qu, Yun Zhai, and Bing An, "The problem of classification in imbalanced data sets in knowledge discovery," in 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), 2010, vol. 9, pp. V9-658-V9-661.

[11] R. Radha and P. Rajendiran, "Using K-Means Clustering Technique to Study of Breast Cancer," in Proceedings of the 2013 International Conference on Information Technology and Applications, 2014, pp. 211–214.

[12] G. G. Sundarkumar, V. Ravi, and V. Siddeshwar, "One-class support vector machine based undersampling: Application to churn prediction and insurance fraud detection," in 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2015, pp. 1–7.

[13] W. Zhang, R. Ramezani, and A. Naeim, "WOTBoost: Weighted Oversampling Technique in Boosting for imbalanced learning," in 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 2523–2531.

[14] G. Hu and Z. Mao, "Bagging Ensemble of SVM Based on Negative Correlation Learning," Proc. - 2009 IEEE Int. Conf. Intell. Comput. Intell. Syst. ICIS 2009, vol. 1, 2009.

[15] F. Bulut, "Heart attack risk detection using Bagging classifier," in 2016 24th Signal Processing and Communication Application Conference (SIU), 2016, pp. 2013–2016.

[16] http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/

[17] X. Zhao, X. Han, W. Su, and Z. Yan, "Time series prediction method based on Convolutional Autoencoder and LSTM," 2019, pp. 5790–5793.

[18] "Prognostics Center of Excellence - Data Repository." [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/. [Accessed: 06-Jan-2021].

[19] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in 2018 Wireless Telecommunications Symposium (WTS), 2018, pp. 1–5.

[20] J.-W. Yang, Y. Lee, and I.-S. Koo, "Convolutional Autoencoder-Based Sensor Fault Classification," 2018, pp. 865–867.

[21] S. J. Kia and G. G. Coghill, "A mapping neural network using unsupervised and supervised training," in IJCNN-91-Seattle International Joint Conference on Neural Networks, 1991, vol. ii, pp. 587–590 vol.2.

[22] Y. C. Du, W. C. Hu, and L. Y. Shyu, "The effect of data reduction by independent component analysis and principal component analysis in hand motion identification," in The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2004, vol. 1, pp. 84–86.

[23] R. Sambasiva Rao, "Mathematical Neural Network (MaNN) Models Part VI: Single-layer perceptron [SLP] and Multi-layer perceptron [MLP] Neural networks in ChEM- Lab," J. Appl. Chem., vol. 3, pp. 2209–2311, 2014.

[24] B. Wen, X. Feng, Q. Yang, and Z. Li, "Seismic facies recognition based on prestack data using two-dimensional convolutional auto-encoder and cluster analysis," in 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP), 2019, pp. 430–434.

[25] L. Vu and Q. U. Nguyen, "An Ensemble of Activation Functions in AutoEncoder Applied to IoT Anomaly Detection," 2019, pp. 534–539.

[26] D. Stursa and P. Dolezel, "Comparison of ReLU and linear saturated activation functions in neural network for universal approximation," in 2019 22nd International Conference on Process Control (PC19), 2019, pp. 146–151.

[27] A. Wiranata, S. A. Wibowo, R. Patmasari, R. Rahmania, and R. Mayasari, "Investigation of Padding Schemes for Faster R-CNN on Vehicle Detection," in 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2018, pp. 208–212.

[28] P. K. Parashiva and A. P. Vinod, "A New Channel Selection Method using Autoencoder for Motor Imagery based Brain Computer Interface," in 2019 IEEE

International Conference on Systems, Man and Cybernetics (SMC), 2019, pp. 3641–3646.

[29] C. Lee, P. Gallagher, and Z. Tu, "Generalizing Pooling Functions in CNNs: Mixed, Gated, and Tree," IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 863–875, 2018.

[30] M. Kolarik, R. Burget, and K. Riha, "Upsampling Algorithms for Autoencoder Segmentation Neural Networks: A Comparison Study," 2019, pp. 1–5.

[31] R. Gu, F. Shen, and Y. Huang, "A parallel computing platform for training large scale neural networks," in Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013, 2013, pp. 376–384.

[32] Y. Ju, X. Wang, and X. Chen, "Research on OMR Recognition Based on Convolutional Neural Network Tensorflow Platform," 2019, pp. 688–691.

[33] K. Komoto, S. Nakatsuka, H. Aizawa, K. Kato, H. Kobayashi, and K. Banno, "A performance evaluation of defect detection by using Denoising AutoEncoder Generative Adversarial Networks," 2018, pp. 1–4.

[34] H. Fu, J. Li, and L. Liang, "Comparison of dimensionality reduction methods for TCM symptom information," 2018, pp. 1861–1865.