Master's Thesis

Detecting Near Miss Incidents in Bicycle Traffic Using Acceleration Sensor Data

Albert Sánchez Fuster

31st March, 2020

Advisors: Prof. Dr.-Ing. David Bermbach

Ahmet-Serdar Karakaya

Technische Universität Berlin
Einstein Center Digital Future

Abstract

Nowadays, with the improvement of the hardware devices and its processing capabilities as well as the big data field, a new promising stage in the Artificial Intelligence is born. Hence, a lot of useful applications involving AI are being developed which enable automatizing activities, patterns and voice recognition, pattern and image classification… Taking all this into consideration, this thesis intended to develop an incident detector and classifier using data collected by the SimRa App. This App records the rides from cyclists using the accelerometer sensor of the smartphone where it runs. First, the dataset will be overviewed to get an insight of how many useful data is available. Then a data extraction and a subsequent signal enhancement, with filtering and other signal processing techniques like ICA, will be performed in order to improve the quality of the signal that will feed the end classifier. Afterwards, a data adaptation and the developing of several classifiers will be done in order to obtain a reliable incident detector and a satisfactory incident classifier. Finally, an evaluation of the full algorithm processing time will be held as a means to know the viability of the deployment in the SimRa smartphone App.

Contents

Chapter 1

Introduction

1.1. Motivation

The SimRa project [1] aims to provide an overview of the bicycle traffic as well as the near miss incidents during the inner-city rides. The collected data will be used and analysed so as to improve city planning and help city councils so as to act in such a way to avoid 'black spots' in the city in order to make it more secure for the cyclists.

1.2. Problem Statement

There are 3 different problems that must be faced in this thesis which will solve the incident detection and classification issue, currently not being precisely considered in the SimRa project.

1. The main problem is the classification of the different type of incidents defined by the project. It is required to isolate the different signal characteristics, which will have noise components, sometimes difficult to filter, for each of the incidents' types. Fulfilling this point will improve the usability as well as the performance of the incidents' classification task.

2. Another big issue that the thesis aims to fix is the precision of the existing labeled data. When users manually label the incidents, it is reasonable to assume that the location of the user generated incident marker is not very accurate. Thus, a solution must be found so as not to take erroneous accelerometer signals.

3. The incident detection algorithm implemented in SimRa does not distinguish between different types of cyclists and their different ways of driving, which can create false positives, hence classification between type of users and a threshold between them should be defined.

Solving the topics listed above will be a significant improvement for the autonomous incident detection and classification which will (1) allow users to label less data than before due to the enhanced accuracy and the lower error rate and (2) obtain more accurate data in order to calculate results with a higher precision in the analysis.

1.3. Goals

The main goals of the project, as stated before, are:

1. Improve the actual incident detection algorithm. The succeed in this point will help improving the dataset, making it more reliable and valuable. Then its data will be more accurate in terms of the classification between incident / no incident detection.

2. Once the improvement in the incident detection algorithm is done, different classification algorithms will be developed in order to evaluate the results when trying to classify between the different types of incidents defined in the SimRa project.

Chapter 2

SimRa Project background

2.1. Project overview

One of the main aims in big cities nowadays is to reduce traffic emissions and try to change its population shifts, based in private vehicles, to a more sustainable transportation methods such as bicycles. Although introducing the bike as a reliable transport in which people feel secure seems to be an easy thing, it requires a good city planning that considers the safety and an overall incidents overview in order to rethink the traffic in some city spots and avoid as much as incidents as possible. Nevertheless, the global overview of all the incidents occurred in a city is difficult to discern due to the fact that in the official statistics only the crashes are pointed out but not the near miss incident which in fact, are important as well in order to know which spots are insecure and where are they.

2.2. Goals of the project

SimRa's goals are to collect data from cyclists using an iOS and Android App (available in both applications stores) in order to target two problems: 1) collect the near crashes that nowadays are being omitted, in order to reflect them in the official statistics, and 2) identify the main bike routes of the cities in order to be able to plan better the general city traffic taking into account the bikes.

2.3. State of the art on incident detection

Currently, SimRa App is equipped with an automated incident detection that tries to reliable identify and detect the incident without any trained data. Several assumptions were made and the algorithm based the incident detection in sudden acceleration peaks. In order to do so and filter poor road conditions, the algorithm group the acceleration values in windows of three seconds, then identify the maximum and minimum value and calculate the difference. Afterwards, it assumes as incidents the two highest variations for each of the accelerometer axis (X, Y and Z). This detection scheme works rather well for cyclists with a ‘relaxed’ style but not for cyclists with more ‘rapid’ cycling style.

Chapter 3

Dataset overview and data extraction

The first stage of the project, in order to finally achieve the main goal, is to generate some kind of reports that could present a summary of the the SimRa's available dataset available in a GitHub Repository[2]. That reports can be of great help due to the fact that a big amount of data is needed in order to feed and train deep learning classifiers such as the one will be used later on in this work. This will allow a clear overview of the available data, and determine the usefulness of the labeled data. In this stage, a Java Application has been developed to extract all the information and summed it all up in 2 Excel files described later.

3.1. Objectives

The goals of this stage are listed below:

- Extract the available dataset of SimRa
- Compact the data
- Summarize the useful labeled data
- Obtain the useful data in an organized way
- Deliver the data to the next process in the project in a concrete format.

3.2. Tasks

So as to achieve the goal of this stage several tasks were defined:

- Java Application development

In order to organize the data and extract the relevant information from it, a Java Application was developed using NetBeans 8.2 IDE. The source code is available on GitHub [3]. The main workflow of this application it's depicted below, which is very simple due to the functionality of the program.

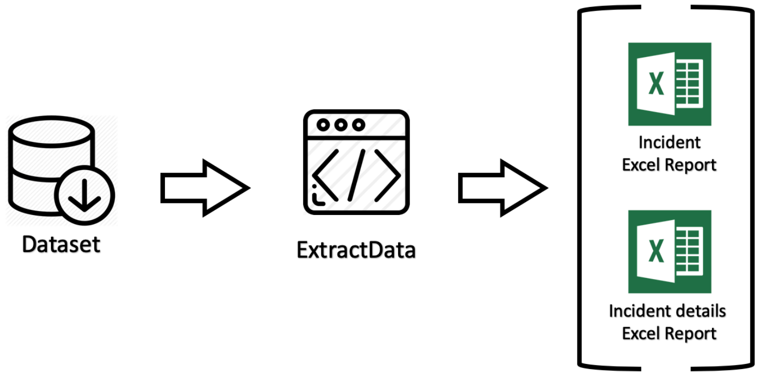


Figure 3.1: ExtractData Application Workflow

- Output excels formatting

  The aim of the ExtractData program is to generate two Excel files that sum up all the available data in the dataset of SimRa. There are two different files which are:

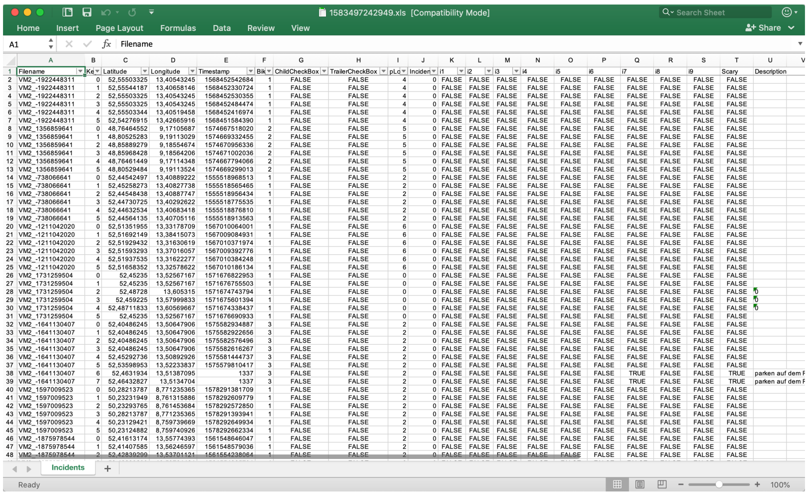
  - Incidents File
  - Detailed Incidents File

Incidents File

The incidents file shows all the recorded incidents and describe the type and other characteristics of them. In this file we have the following parameters available:

- **Filename:** Filename where incident is found
- **Key:** Incident number in the file
- **Latitude:** Latitude coordinates

- **Longitude:** Longitude coordinates
- **Timestamp:** Time reference
- **Bike:** Bike type
- **ChildCheckBox:** Child during the ride?
- **TrailerCheckBox:** Trailer during the ride?
- **pLoc:** Phone Location
- **Incident:** Incident Type
- **i1-i9:** Participants involved in the incident
- **Scary:** Was the incident scary?



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Filename	Ke	Latitude	Longitude	Timestamp	Bik	ChildCheckBox	TrailerCheckBox	pLo	Inciden	i1	i2	i3	i4	i5	i6	i7	i8	i9	Scary	Description	
2	VM2_-1922448311	0	52,55503325	13,40543245	1568452542684	1	FALSE	FALSE	4	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
3	VM2_-1922448311	1	52,55544187	13,40658146	1568452330724	1	FALSE	FALSE	4	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
4	VM2_-1922448311	2	52,55503325	13,40543245	1568452530355	1	FALSE	FALSE	4	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
5	VM2_-1922448311	3	52,55503325	13,40543245	1568452484474	1	FALSE	FALSE	4	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
6	VM2_-1922448311	4	52,55503344	13,40519458	1568452416974	1	FALSE	FALSE	4	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
7	VM2_-1922448311	5	52,54276915	13,42665916	1568451584390	1	FALSE	FALSE	4	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
8	VM2_1356859641	0	48,76464552	9,17105687	1574667518020	2	FALSE	FALSE	5	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
9	VM2_1356859641	1	48,80525283	9,19113029	1574669332455	2	FALSE	FALSE	5	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
10	VM2_1356859641	2	48,85889279	9,18554674	1574670956336	2	FALSE	FALSE	5	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
11	VM2_1356859641	3	48,85968428	9,18564206	1574671002036	2	FALSE	FALSE	5	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
12	VM2_1356859641	4	48,76461449	9,17114348	1574667794066	2	FALSE	FALSE	5	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
13	VM2_1356859641	5	48,80529484	9,19113524	1574669299013	2	FALSE	FALSE	5	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
14	VM2_-738066641	0	52,44542497	13,40889222	1555518968513	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
15	VM2_-738066641	1	52,45258273	13,40827738	1555518565465	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
16	VM2_-738066641	2	52,44548438	13,40887747	1555518956434	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
17	VM2_-738066641	3	52,44730725	13,40292622	1555518775535	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
18	VM2_-738066641	4	52,44632534	13,40683418	1555518876810	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
19	VM2_-738066641	5	52,44564135	13,40705116	1555518913563	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
20	VM2_-1211042020	0	52,51351955	13,33178709	1567010064001	1	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
21	VM2_-1211042020	1	52,51692149	13,38415073	1567009084931	1	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
22	VM2_-1211042020	2	52,51929432	13,31630619	1567010371974	1	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
23	VM2_-1211042020	3	52,51593293	13,37016057	1567009392776	1	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
24	VM2_-1211042020	4	52,51937535	13,31622277	1567010384248	1	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
25	VM2_-1211042020	5	52,51658352	13,32578622	1567010186134	1	FALSE	FALSE	6	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
26	VM2_1731259504	0	52,45235	13,52567167	1571676822953	1	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
27	VM2_1731259504	1	52,45235	13,52567167	1571676755503	1	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
28	VM2_1731259504	2	52,48728	13,605315	1571674743794	1	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	
29	VM2_1731259504	3	52,459225	13,57999833	1571675601394	1	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	
30	VM2_1731259504	4	52,48711833	13,60569667	1571674338437	1	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	0	
31	VM2_1731259504	5	52,45235	13,52567167	1571676690933	1	FALSE	FALSE	0	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
32	VM2_-1641130407	0	52,40486245	13,50647906	1575582934887	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
33	VM2_-1641130407	1	52,40486245	13,50647906	1575582922656	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
34	VM2_-1641130407	2	52,40486245	13,50647906	1575582576496	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
35	VM2_-1641130407	3	52,40486245	13,50647906	1575582616267	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
36	VM2_-1641130407	4	52,45292736	13,50892926	1575581444737	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
37	VM2_-1641130407	5	52,53598953	13,52233837	1575579810417	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
38	VM2_-1641130407	6	52,4631934	13,51387095	1337	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	parken auf dem F	
39	VM2_-1641130407	7	52,46432827	13,5134704	1337	3	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	parken auf dem F	
40	VM2_1597009523	0	50,28213787	8,771235365	1578291381709	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
41	VM2_1597009523	1	50,23231949	8,761315886	1578292609779	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
42	VM2_1597009523	2	50,23293765	8,761453684	1578292572850	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
43	VM2_1597009523	3	50,28213787	8,771235365	1578291393941	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
44	VM2_1597009523	4	50,23129421	8,759739669	1578292649934	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
45	VM2_1597009523	5	50,23124882	8,759740926	1578292662334	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
46	VM2_-1875978544	0	52,41613174	13,55774393	1561548646047	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
47	VM2_-1875978544	1	52,41407585	13,56246597	1561548579036	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		
48	VM2_-1875978544	2	52,42839299	13,53701121	1561554238064	1	FALSE	FALSE	2	0	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE		

Figure 3.2: Incident Excel File screenshot

Detailed Incidents File

The detailed incidents file shows the different incidents. There are 8 worksheets, one for each type of incident, and inside there are all the incidents which can be distinguished by the 'Key' value. In this file we have the following parameters available:

- **DS_Name:** Filename where incident is found
- **Key:** Incident number in the file
- **Type:** Incident type (redundant)

- **Latitude:** Latitude coordinates
- **Longitude:** Longitude coordinates
- **Acc_X:** Accelerometer X values
- **Acc_Y:** Accelerometer Y values
- **Acc_Z:** Accelerometer Z values
- **Timestamp:** Time reference
- **Acc_68:** Radius of 68% of confidence.
- **Gyr_A:** Gyroscope A values
- **Gyr_B:** Gyroscope B values
- **Gyr_C:** Gyroscope C values

3.3. Results

Once the data extraction was done, 65.958 labeled incidents were found in the dataset. The results showed clearly that most of the incidents detected by the previous algorithm, concretely 59.377 out of 65.958 (90%), were false positives that is, useless incidents. The remaining 10% of the dataset could be divided into true positives and false negatives. The true positives, a total of 3.143 (5%), are incidents that are detected by the algorithm and classified by the user between type 1 and type 8. The false negatives, a total of 3.438 (5%), are incidents that haven't been detected by the algorithm but the users tagged them manually.

This manually tagged incidents face one problem which is the accuracy of the tag placed by the user. This accuracy will be evaluated in section 7.3 once the classifier is built but is briefly explained below.

The test taking 2 different datasets aimed to clarify if adding the false negative incidents (also known as user tagged incidents) to the useful dataset (the true positives) improve or not the overall evaluation metrics when performing the classification. The result was that using both types of incidents, user tagged and automatically tagged incidents, decrease the results of the classifier so, it can be finally conclude that the useful dataset that will be used in order to develop the final steps of the project will be the true positives dataset, that is a 5% of the hole dataset and corresponds to 3.143 incidents.

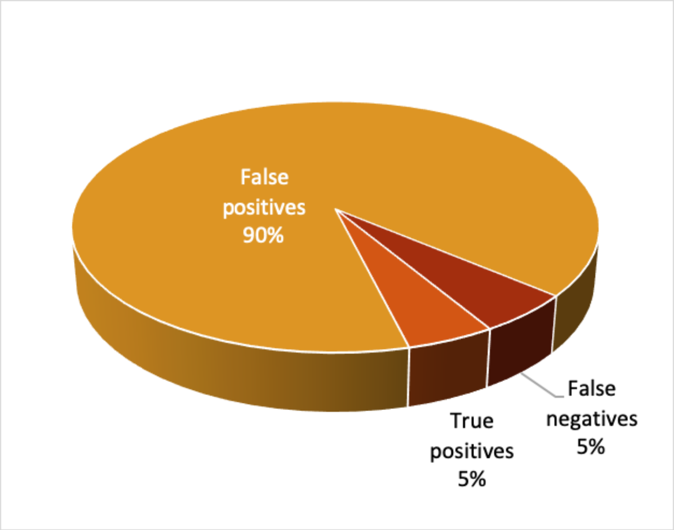


Figure 3.3: Available dataset overview classification

In addition, in the following figure can be seen the different types of incidents and the percentage they have among the others in the useful 5% of the dataset.

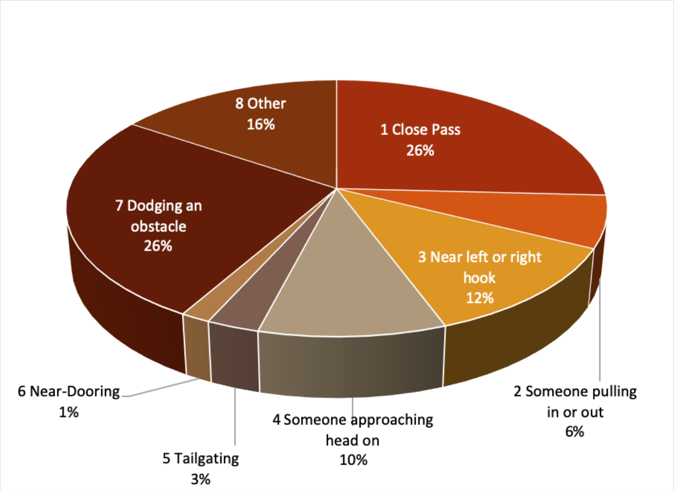


Figure 3.4: True positives dataset overview – Incident types

Chapter 4

Accelerometer data enhancement

In this stage of the project several tests and trials have been performed with the aim of knowing the quality of the data recorded by the SimRa App. However, the evaluation of the signal quality was not the only goal set in this stage. Here, a frecuencial analysis will be performed as well as the application of several signal processing procedures. After the frecuencial analysis is done, several filter designs will be made (low, high and band pass filters) in order to analyze the response of the dataset signals when applying these filters. Afterwards, the independent component analysis technique will be applied.

With all these tests and techniques, a trial to enhance the signals from the dataset is done, in order to improve the input signals before the classification task.

All the Matlab code used to perform this tests and trials is available in a GitHub Repository [4]

4.1. Evaluation of the accelerometer data

In this stage, 2 different tests have been performed: The No movement test and the 3-axis test.

4.1.1 No Movement test

In this test, a recording of a 5 seconds ride with SimRa App was done. During these 5 seconds the Smartphone was quiet on a table.

The goal of this test was to see the raw data of a ride and evaluate its quality in a representation of the data using Matlab. As we can see in figure 4, where the 3 axis accelerometer signals are represented, the recorded signal take values within a range of $6 \cdot 10^{-3}$ m/s^2.

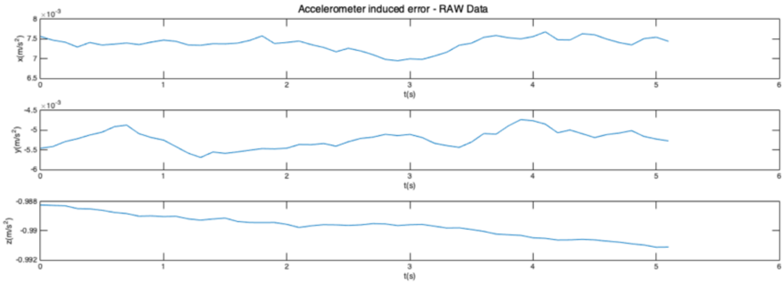


Figure 4.1: 3-Axes accelerometer signals in the No movement test

4.1.2 3-Axes test

In this test, 3 recordings of 5 seconds rides with SimRa App were done. In each of these recordings, one of the accelerometer axes was tested moving the smartphone according to that axis.

The scope of this test was to see the raw data of a movement and evaluate its quality in a representation of the data using Matlab. In figures 4.2, 4.3 and 4.4, it can be seen the evolution of the 3-Axes accelerometer values for each of the recordings depending on the axis variation.

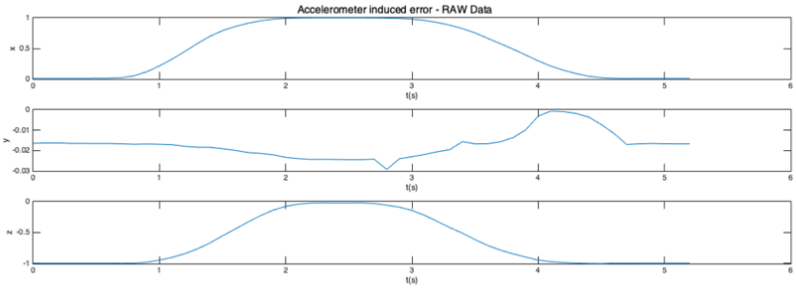


Figure 4.2: 3-Axes accelerometer signals in X-Axis variation test

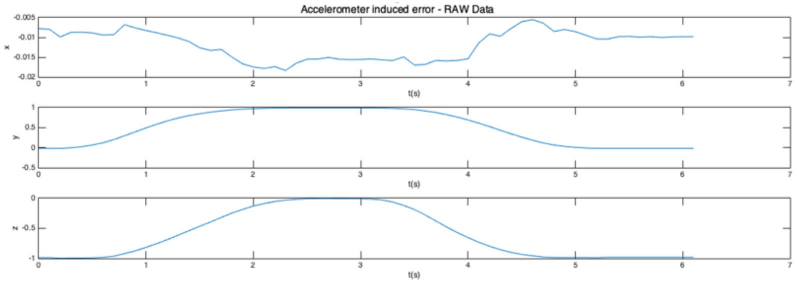


Figure 4.3: 3-Axes accelerometer signals in Y-Axis variation test

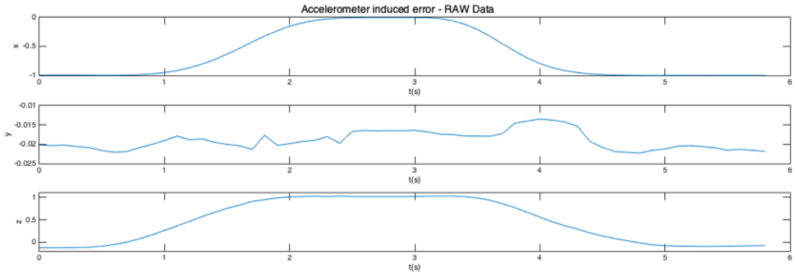


Figure 4.4: 3-Axes accelerometer signals in Z-Axis variation test

4.2. Frecuencial analysis

In this stage, N signals for each of the incident type were extracted from the dataset and adapted to be processed by Matlab. In the figures 4.5 and 4.6 can be seen the raw representation of one signal for each type of incident and the associated Fourier transform for the 3-axes accelerometer values.

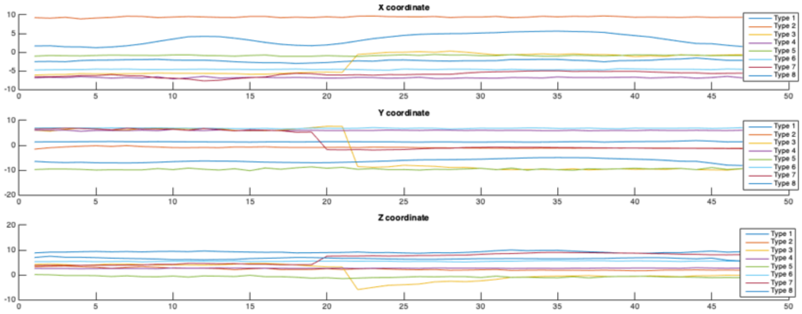


Figure 4.5: Raw signals representation (1 signal per incident type)

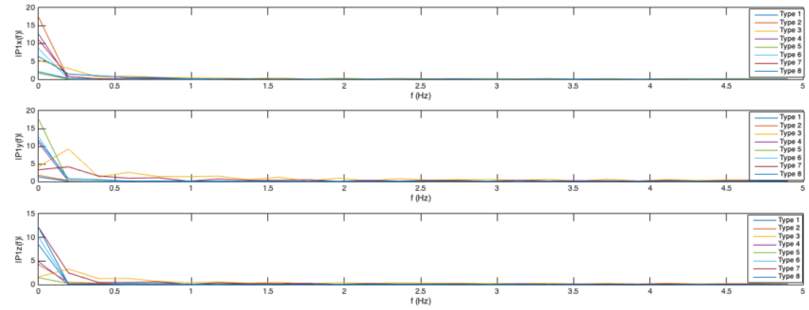


Figure 4.6: Fourier transform representation of signals of figure 4.5

In the representation of the Fourier transform can be seen the frequency components of the different type of signals for the X, Y and Z axis of the accelerometer. It can be observed that the main frequency components are in the lower bands. This observation permit to see that high pass filter won't be useful and will end up filtering nearly the hole signal. In the filter section 4.3 a test will be done in order to illustrate this.

4.3. Filter designs

In this stage, some signal filtering was applied to the dataset signals in order to make an analysis about the impact of each of the 4 designed filters. Based on [5], an analysis with different filters was done and the 4 filters used in this experiment are described below:

4.3.1 High pass filter

The first filter was designed in order to delete the part of the spectrum below 0.5 Hz. This decision was made considering figure 4.6 where it can be observed that most of the energy signal is below 0.5 Hz. The result of this filter will be the nearly complete deletion of the signal. Consequently, this filter is not useful for the data used in this project.

The designed filter was a Finite Impulse Response Digital Filter, with a length of 191 samples. The design method used to obtain the filter was the Kaiser window and the design specifications were the following:

- Sample Rate: 10 Hz
- Passband edge: 0.5 Hz
- Passband Ripple: 0.5 dB
- Stopband edge: 0.25 Hz
- Stopband attenuation: 70 dB

In figure 4.7, the resulting Magnitude response and the Phase response of the designed filter can be observed.

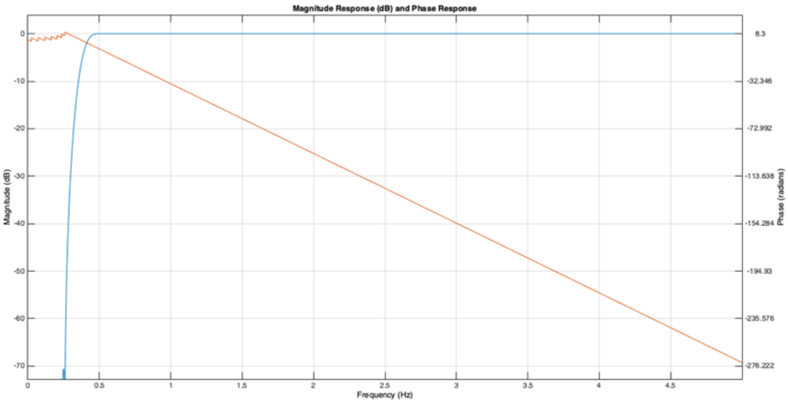


Figure 4.7: Magnitude and Phase response of the designed High Pass Filter

4.3.2 Band pass filter

The second filter was designed in order to delete the part of the spectrum below 0.5 Hz and above 3 Hz. This filter, a priori was useless due to the spectrum of the different signals showed in figure 4.6 . This is just a test in order to show that band pass filters in the concrete case of our project are not useful at all and that the output of this filter will be some signals that will end up being useless for our next project steps.

The designed filter was a Finite Impulse Response Digital Filter, with a length of 21 samples. The design method used to obtain the filter was hamming window and the design specifications were the following:

- Sample Rate: 10 Hz
- Filter order: 20
- First Cutoff frequency: 0.5 Hz
- Second Cutoff frequency: 3 Hz

In figure 4.8, the resulting Magnitude response and the Phase response of the designed filter can be observed.

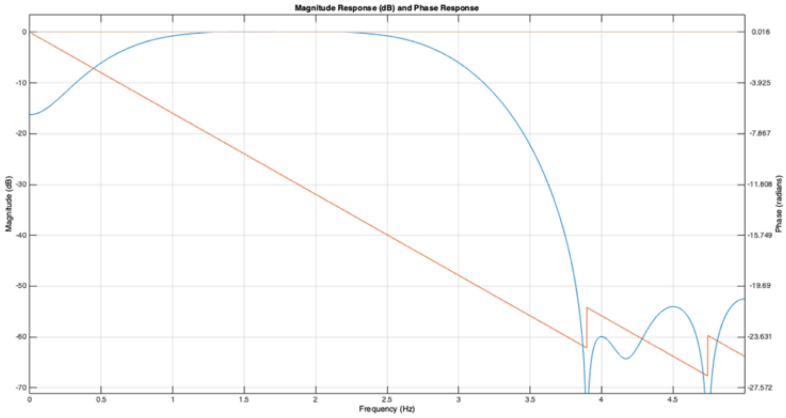


Figure 4.8: Magnitude and Phase response of the designed Band Pass Filter

4.3.3 Low pass filter 1

The third filter was designed in order to delete the part of the spectrum above 4 Hz. This decision was made considering the figure 4.6 where it can be observed that most of the energy signal is in the lower bands.

The designed filter was a Finite Impulse Response Digital Filter, with a length of 21 samples. The design method used to obtain the filter was the Kaiser window and the design specifications were the following:

- Sample Rate: 10 Hz
- Filter order: 20
- Cutoff frequency: 4 Hz

In figure 4.9, the resulting Magnitude response and the Phase response of the designed filter can be observed.

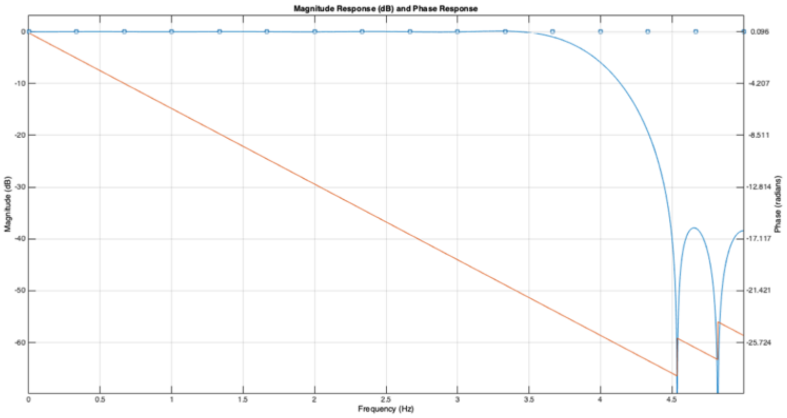


Figure 4.9: Magnitude and Phase response of the designed Low Pass Filter 1

4.3.4 Low pass filter 2

The fourth filter was designed in order to delete the part of the spectrum above 2.5 Hz. This filter is just a variation in order to observe the differences between the previous low pass filter with cutoff frequency of 4 Hz.

The designed filter was a Finite Impulse Response Digital Filter, with a length of 21 samples. The design method used to obtain the filter was the Kaiser window and the design specifications were the following:

- Sample Rate: 10 Hz
- Filter order: 20
- Cutoff frequency: 2.5 Hz

In figure 4.10, the resulting Magnitude response and the Phase response of the designed filter can be observed.

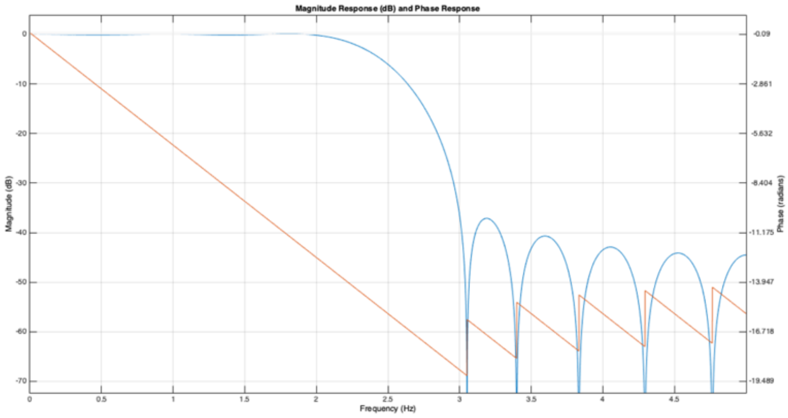


Figure 4.10: Magnitude and Phase response of the designed Low Pass Filter 2

4.4. Independent component analysis

In this stage, the Independent Component Analysis was performed. The aim of this procedure is decomposing mixed signals into independent sub-parts as described in [6]. In the case of this project, the real meaning of decomposing mixed signals is trying to separate the useful signal that should contain the incident itself, from the mixed signal with the noise of pedalling and the ride.

4.4.1 Theory

ICA analysis is based on assuming that there are m observations $\boldsymbol{x}$ which are linear mixtures of n independent components $\boldsymbol{s}$ as showed in (4.3).

$$\mathbf{x} = (x_1, ..., x_m)^T \tag{4.1}$$

$$\mathbf{s} = (s_1, ..., s_n)^T \tag{4.2}$$

$$x_i = a_{i,1} \cdot s_1 + ... + a_{i,n} \cdot s_n \tag{4.3}$$

Each of this component is formed by a sum of mixture coefficients $a_{i,k}$ and the original source signals s_k that could be represented as in (4.4)

$$\mathbf{x} = \sum_{k=1}^{n} \mathbf{a}_k \cdot s_k = \mathbf{A} \cdot \mathbf{s} \tag{4.4}$$

where

$$\mathbf{a}_k = (a_{1,k}, ..., a_{m,k})^T, \tag{4.5a}$$

$$\mathbf{A} = (a_1, ..., a_n) \tag{4.5b}$$

Defined equation (4.4), the algorithm tries to estimate both parameters $\boldsymbol{A}$, the matrix of mixture coefficients, and $\boldsymbol{s}$, the n original source signals The estimation of these two parameters is done by recursively calculate the $\boldsymbol{w}$ vectors in order to maximize the non-gaussianity, given a cost function, of the equation (4.6).

$$s_k = w^T \cdot x \tag{4.6}$$

Finally, the original sources signals can be recovered applying (4.7), that is inverting the matrix of mixture coefficients A and multiplying it by the observed signals x:

$$\boldsymbol{s} = \boldsymbol{W} \cdot \boldsymbol{x} \tag{4.7}$$

where

$$\boldsymbol{W} = \boldsymbol{A}^{-1} \tag{4.8a}$$

4.4.2 Adaptation to project scope

In order to use this technique, a second signal for every dataset signal was needed in order to apply ICA. Then, with this second signal there will be 1) an incident signal plus a normal ride signal, coming from the dataset, and 2) a test signal that supposed to simulate a normal ride signal without incident. Therefore, a ‘test’ signal was recorded in an experimental bike ride of 10 seconds. This bike ride was done under carefully conditions, riding in a straight way and in a good condition pavement street.

A schema of the intended experiment is depicted in figure 4.11

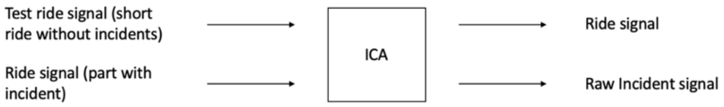

Test ride signal (short ride without incidents)
Ride signal (part with incident)
ICA
Ride signal
Raw Incident signal

Figure 4.11: ICA Analysis test procedure overview

4.4.3 Tests

Using the available Matlab Package [7], an adaptation code to use it was developed and several tests were performed. These tests were developed in order to obtain signals, as previously depicted, with only the useful incident signal information. In this section, some examples of the performed tests in every signal of the dataset are visualised.

For instance, in figure 4.12, it can be observed that the resulting output signals of ICA seem that could have been properly decomposed. On the contrary, in figure 4.13, the resulting signal coming from the decomposition seem that the ICA analysis is not useful for this case.

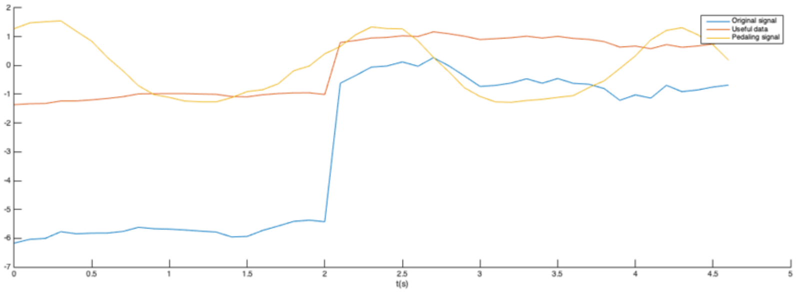


Figure 4.12: ICA Analysis resulting signals (Applied to signal of incident type 3)

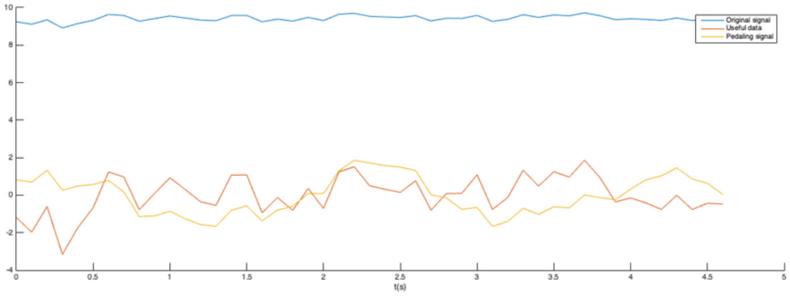


Figure 4.13: ICA Analysis resulting signals (Applied to signal of incident type 2)

4.5. Decision parameters

In this stage, and after having performed all the previous steps in data enhancement for the accelerometer data, some parameters have to be defined in order to set the inputs for the classification algorithm that will be afterwards explained in chapter 6.

Before developing the program that will adapt the dataset and transform it to the desired format, which is later explained in Adapt Dataset section, some previous studies were made in Matlab. These studies were made in order to have some visual overview of the data and how these parameters behave depending on the incident type.

After making the tests in Matlab environment and having performed some research on which parameters have bigger influence when trying to classify data based on accelerometer signals ([5], [8] and [9]) the final parameters where set. The final parameters were the following ones:

- Speed
- Mean of accelerometer values in X, Y and Z axes
- Standard deviation of accelerometer values in X, Y and Z axes
- Signal Magnitude Area (SMA)

$$\boldsymbol{SMA = \frac{1}{i}(\sum_{u=1}^{i} |x_u| + \sum_{u=1}^{i} |y_u| + \sum_{u=1}^{i} |z_u|)} \tag{4.9}$$

- Mean of Signal Vector Magnitude (SVM)

$$\boldsymbol{SVM = \frac{1}{N}\sum_{i=1}^{N}\sqrt{x_i{}^2 + y_i{}^2 + z_i{}^2}} \tag{4.10}$$

- Entropy of accelerometer values in X,Y and Z
- Phone Location
- Bike type

Below can be found some examples of the final plots obtained in Matlab that show some of these decision parameters. Each of the circles show the value of the calculate magnitude for the specific ride. Moreover, these circles have the color of the incident type they pertain and the line plotted for each of the incident types show the mean value of the magnitude for the specific incident type.

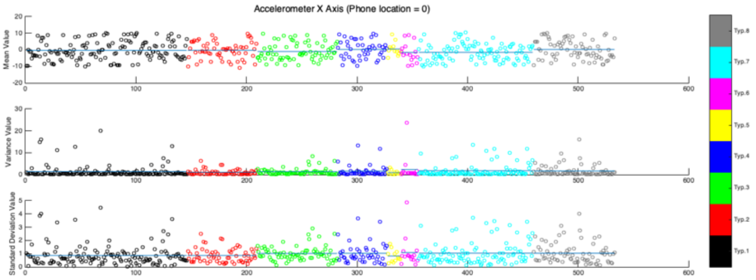


Figure 4.14: Matlab plot for parameters Mean, Variance and Standard Deviation of Accelerometer X Axis and incidents with phone location = 0 (Pocket)

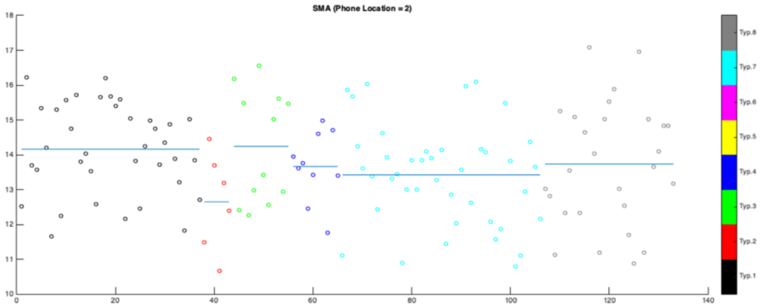


Figure 4.15: Matlab plot for decision parameter SMA and incidents with the phone location = 2 (Jacket pocket)

Chapter 5

Adapt Database

In this stage, a Java application available in a GitHub Repository [10], was built in order to transform the SimRa dataset with the accelerometer signals and metadata from the involved incident into a set of variables that could be used as inputs of the Neural Network that will be used to classify the different type of incidents. The main workflow of the program is to extract the rides of the dataset that have incidents associated with them. After obtaining all the rides, the program transforms the ride into N-Windowed Rides and for each of these windowed rides (with windows of 6 seconds by default) it obtains the decision parameters explained in the previous section. So, 1 Ride becomes N windowed rides, and for one windowed ride, 14 decision parameters are extract.

In the following figure, the main workflow of the application can be observed.

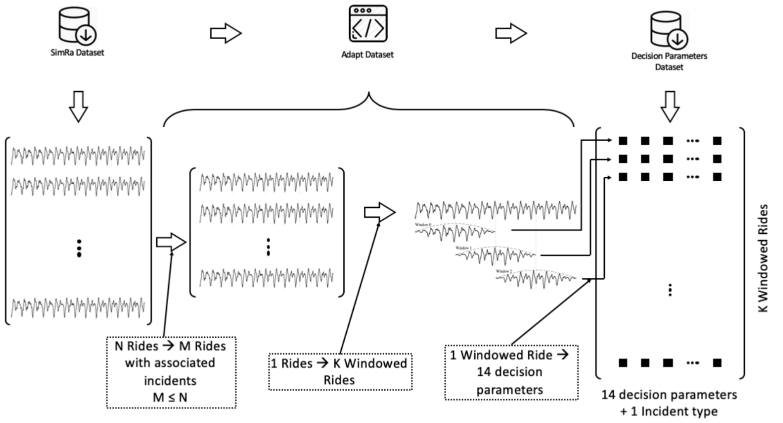

SimRa Dataset
Adapt Dataset
Decision Parameters Dataset
N Rides → M Rides with associated incidents M ≤ N
1 Rides → K Windowed Rides
1 Windowed Ride → 14 decision parameters
K Windowed Rides
14 decision parameters + 1 Incident type

Figure 5.1: Adapt Database workflow

Chapter 6

Machine learning applied to accelerometer data

6.1. Related work

Smartphone accelerometers are widely used in Artificial Intelligence applications. However, the most common goals are focused on Human Activity Recognition, also known as HAR detectors where some examples can be found in [9], [11], [12], [13], [14] ... These HAR detectors are mainly build using accelerometer data as input data and they are implemented using different techniques depending on the final goal. The most common implementations are based on Artificial Neural Networks with structure of Multi-Layer Perceptrons but there are others implemented using Recurrent Neural Networks such as Long-Short Term Memory (LSTM) [12] or Convolutional Neural Networks (CNN) [15]. Just as an insight of different scope current researches using this type of technologies, there are some focused on road damages [16] and others focused on accident (but not incident) detection [17].

6.2. Deep Learning - Artificial neural network

6.2.1 Introduction to Deep Learning

Deep Learning is a subset of Machine Learning based on Artificial Neural Networks (ANN) that tries to imitate the biological neural networks that constitute brains.

These systems are characterised by its learning ability that can be done by different approaches: supervised, unsupervised or mixed learning. Moreover, unlike Machine Learning algorithms that need expertise in order to extract the features that will be used by the classification algorithm, Deep Learning includes this feature extraction inside the algorithm itself, allowing to omit the expertise in the concrete problem. These algorithms are also characterised by the problem that its implementation wants to solve for instance, there are classification, regression, clustering, among others techniques.

The approach that this project will focus and use will be the supervised learning and the goal of this project is to classify the near-miss incidents into the different types of incidents defined by the SimRa project. Therefore, classification algorithms have to be considered and concretely, a Multi-Layer Perceptron will be implemented.

6.2.2 Multi-Layer Perceptron

A Multi-Layer Perceptron or MLP is a type of Deep Learning algorithm. It is used for classification purposes and the learning is done by supervised methods. This classification algorithm is composed of 3 layers: these are the input, the hidden and output layer. The input layer is where the primary features of the data needed by the Neural Network are introduced. These features are characteristics of the data we want to classify. In this project these features are described in section 4.5. Depending on the chosen features, the performance of the network will vary due to the importance of the inputs while learning the characteristics to perform the classification. The hidden layer can be in fact more than one layer. Depending on the problem it is faced, more or less layers will be needed. More layers mean more computational complexity as well as, generally, better performance of the classification algorithm. The output layer is the final layer of the network and it has the number of nodes equal to the number of types that is needed to classify. Each layer is formed by neurons (also known as Perceptrons) whose function is to perform a weighted sum of the inputs, add a bias term to the result, and finally pass the value to a non-linear activation function.

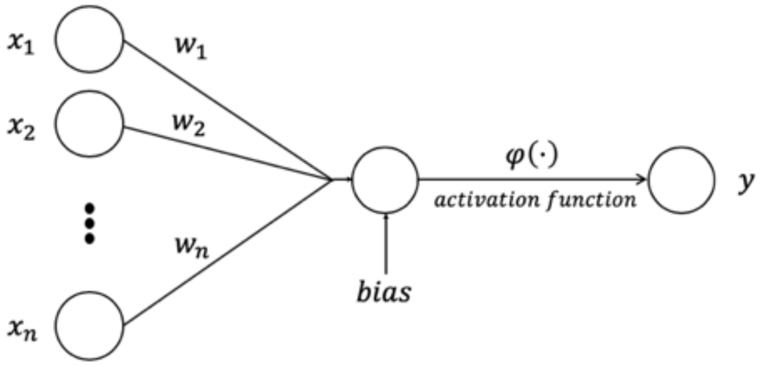


Figure 6.1: Perceptron Scheme

The number of hidden layers as well as the number of neutrons of each of the layers is determined empirically. There is no theory on how many neutrons does each layer needs and how many hidden layers are necessary to solve a specific problem.

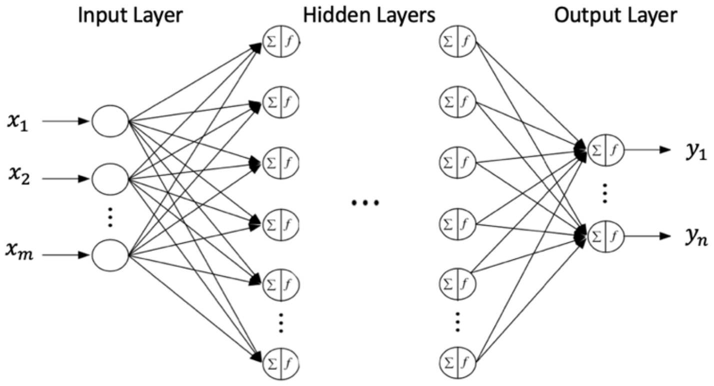


Figure 6.2: Multi-Layer Perceptron Scheme

Once the structure of the different layers is defined, also known as model, there are two important stages to be defined when working with these types of algorithms: the training stage and the test or evaluation stage.

The training stage is when the coefficients of each perceptron are updated and optimized. This procedure will use the training data to improve the model's ability to predict. This will be achieved using the back propagation algorithm.

The test or evaluation stage is done once the training stage is finished. In this stage the results about accuracy, precision, recall, among other, are calculated but the model is not modified anymore.

6.2.3 Hyperparameters of a Multi-Layer Perceptron

The hyperparameters of an MLP Neural Network are those parameters that are need to be set before the training phase starts. These parameters affect directly to the overall performance of the Neural Network. However, there are no theoretical definitions on how these parameters have to be set up and the tuning of them is made through trial and error or optimization procedures that mainly run simulations of the final performance of the network when setting the parameters to different values. These optimization procedures are for instance, Grid Search, Random Search, among others.

The most commonly tuned hyperparameters in an MLP Neural Network are:

- Learning Rate
- Number of Epochs
- Batch Size
- Number of layers
- Number of neutrons per layer

6.2.4 Evaluation Metrics for a Neural Network

Once the training has been done, some evaluation metrics have to be set in order to overview the performance of the designed and trained model. These evaluation metrics are calculated from the confusion matrix that is the result of the test phase (after the training phase).

Confusion Matrix

The confusion matrix shows how the classification has been made. Knowing how the inputs should have been classified it depicts the real classification made by the algorithm. Usually the matrix has the input classes as rows, and the output classes as the columns. A binary classification confusion matrix is depicted afterwards in order to set an example and clarify the concept of a confusion matrix:

0	1	
A	B	0=0
C	D	1=1

A: Number of samples of type 0 that have been classified as type 0

B: Number of samples of type 0 that have been classified as type 1

C: Number of samples of type 1 that have been classified as type 0

D: Number of samples of type 1 that have been classified as type 1

Once the concept of confusion matrix is clear, the metrics itself are defined. In this project, as well as the great majority of classification projects solved with Deep Learning, 4 metrics are set in order to know the performance of the model. These evaluation metrics are the following:

- Accuracy
- Precision
- Recall
- F1 Score

Accuracy

The accuracy is the ratio of good predictions to total predictions:

$$\boldsymbol{\begin{pmatrix} a & b \\ c & d \end{pmatrix} \rightarrow Accuracy = \frac{a+d}{a+b+c+d}} \tag{6.1}$$

Example:

$$\begin{pmatrix} \mathbf{10} & \mathbf{1} \\ \mathbf{2} & \mathbf{7} \end{pmatrix} \rightarrow \boldsymbol{Accuracy} = \frac{\mathbf{10+7}}{\mathbf{10+1+2+7}} = \mathbf{0.85} \tag{6.2}$$

Precision

The precision is the mean of the ratio of good predictions to total positive predictions for each of the classes:

$$\begin{pmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{pmatrix} \rightarrow \boldsymbol{Precision} = \frac{\frac{\boldsymbol{a}}{\boldsymbol{a+c}} + \frac{\boldsymbol{d}}{\boldsymbol{b+d}}}{\mathbf{2}} \tag{6.3}$$

Example:

$$\begin{pmatrix} \mathbf{10} & \mathbf{1} \\ \mathbf{2} & \mathbf{7} \end{pmatrix} \rightarrow \boldsymbol{Precision} = \frac{\frac{\mathbf{10}}{\mathbf{10+2}} + \frac{\mathbf{7}}{\mathbf{1+7}}}{\mathbf{2}} = \mathbf{0.8542} \tag{6.4}$$

Recall

The recall is the mean of the ratio of good predictions to total positive and negative predictions for each of the classes:

$$\begin{pmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{pmatrix} \rightarrow \boldsymbol{Recall} = \frac{\frac{\boldsymbol{a}}{\boldsymbol{a+b}} + \frac{\boldsymbol{d}}{\boldsymbol{c+d}}}{\mathbf{2}} \tag{6.5}$$

Example:

$$\begin{pmatrix} \mathbf{10} & \mathbf{1} \\ \mathbf{2} & \mathbf{7} \end{pmatrix} \rightarrow \boldsymbol{Recall} = \frac{\frac{\mathbf{10}}{\mathbf{10+1}} + \frac{\mathbf{7}}{\mathbf{2+7}}}{\mathbf{2}} = \mathbf{0.8434} \tag{6.6}$$

F1 Score

The F1 Score is the weighted average of Precision and Recall:

$$\boldsymbol{F1Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}} \tag{6.7}$$

6.3. Development

In the following sections the technical development of the final Neural network algorithm is explained. Additionally, all the developments can be found in GitHub repositories detailed in following sections.

6.3.1 DeepLearning4J libraries

In order to develop the Neural Network based on the Multi-Layer Perceptron previously described, some libraries have to be used. The election of the tools was based on the fact that, all this project pretends to be executed and deployed in a Smartphone app. Taking this into account, the libraries developed by DeepLearning4J [18] (or DL4J) was chosen. This set of libraries allow an easy implementation of the Multi-Layer perceptron algorithm, multiple extra features, like Optimization or Data transformation, and an easy integration with a Smartphone App.

6.3.2 Dataset transformation

In order to feed the Neural network described later in 3.5.4.3. some transformation of the dataset has to be done. These adaptations will be performed by the DataVec library available in the DL4J, concretely with the Schema and TransformProcess classes. This dataset transformation allows to, among other things, remove input variables coming from the adapted dataset, in order to do some performance tests depending on the inputs, and encode categorical variables such as Bike Type and

Phone Location using One-Hot encoding, which is recommendable when working with Neural Networks. This dataset transformation will generate a total amount of 28 inputs maximum. This number is obtained from the 12 decimal inputs (1 speed, 3 accelerometer means, 3 accelerometer standard deviation, 1 SMA, 1 SVM mean and 3 accelerometer entropies) plus the 'Bike Type' input that will generate 7 and the 'Phone location' that will generate 5 (due to One-Hot encoding).

6.3.3 Neural Network based on Multi-Layer Perceptron

As stated in previous chapters, a Deep Learning algorithm is going to be developed in order to solve the classification problem that faces this project. Concretely, an Artificial Neural Network (ANN) configured as a Multi-Layer Perceptron (MLP) is going to be programmed. The neural network development is done using different approaches:

- The first approach splits the programming of the neural network into two stages: the first is the model definition and optimisation to choose the hyperparameters, and the second is the training and the testing of the Neural Network model obtained from the previous stage.
- The second approach is using raw definition of the model with the training and testing phase in the same program.

Approach 1: Model definition with optimization

- Stage I: Model Definition and Optimization (using Arbiter)

Using the Arbiter Library, a Neural Network architecture with the possibility to run an optimization process was built. The configuration of the model that will be afterwards optimized is the following:

- Learning Rate = 0.1
- Input layer of 28 inputs

- 5 Hidden layers (the neurons in each layer will be determined by the optimisation procedure)
- Output layer of 9 outputs (3 in case of ternary classification and 2 in case of binary classification)
- Input Layer and Hidden Layers will use a Hyperbolic Tangent activation function
- Output Layer will use a SoftMax activation function
- Loss function will be calculated using a negative log likelihood function
- Additionally, a L2 regularisation method is used to improve overfitting issues.

With this model structure, an optimisation procedure using Random Search generator will be performed. However, due to the lack of simulation environment, some stopping conditions have to be set in order not to overload the CPU of the computer where the optimisation procedure take place. The conditions set were: 10 different candidates generation or maximum running time of 120 minutes. Also, in order to perform a fair optimisation procedure while running different tests with different datasets, a maximum of 1000 iterations where set. This will be accomplished by defining the number of epochs as a function of total number of batches of the dataset and the batch size.

$$\boldsymbol{Number\ of\ epochs = \frac{number\ of\ iterations}{\lceil \frac{total\ batches}{batch\ size} \rceil} = \frac{100}{\lceil \frac{total\ batches}{batch\ size} \rceil}} \tag{6.8}$$

This optimisation procedure will end up generating a JSON file that will contain the model configuration after the optimised parameters are found. This file will be loaded in the next phase in order to train and test it. The optimisation procedure development can be found in this GitHub Repository [19].

- Stage II: Training and testing

After the previous phase, a new program is defined. In it, a data transformation is done to the dataset. Afterwards, a data normalisation is performed using a class called *NormalizerStandardize()* that will, as it name says, normalise the input data to have 0 mean and a variance equal to 1. This normalisation process is important to prevent some input data, that its values are higher than others (for instance speed vs standard deviation of accelerometer readings), have more weight once the internal coefficients are being updated. With this procedure we ensure that all inputs have the same weight in the system. Finally, the model configuration obtained in a JSON file is loaded and the training phase is started.

The principal characteristics of the training phase are the following:

- Splitting dataset: 70% of the dataset used for training – 30% used for testing.
- Number of epochs: 2000.
- Batch size: variable depending on the test

The training ant testing development can be found in this GitHub Repository [20].

Approach 2: Raw definition of the model

Due to the CPU requirements of the optimisation procedure, a manual definition of the model is done. This approach can be found in this GitHub Repository [21] and has the following configuration:

- Learning Rate = 0.1
- Input layer of 28 inputs and output layer of 1 output
- 5 Hidden layers (2000 neurons/layer)
- Input Layer and Hidden Layers will use a Rectified Linear Unit (ReLU) activation function
- Output Layer will use a Sigmoid activation function
- Loss function will be calculated using a Cross Entropy function for a Binary Classification

- Additionally, a L2 regularisation method is used to improve overfitting issues.

Additionally, in a branch called *earlystopping* in the same repository, a version of the same neural network adding an early stopping mechanism can be found.

Chapter 7

Tests and results

7.1. Batch size in optimization stage test

A Neural Network is composed by several hyperparameters that must be tuned in order to obtain good performance. One of these parameters is the batch size that defines the number of samples that will be propagated through the network before updating the coefficients inside it (one forward/backward pass). Another one is the number of epochs that are used to train the Network. This define the number of times that the hole dataset will be used. One epoch will be one forward pass and one backward pass of all the examples in the dataset.

For instance, if the dataset that contains 1000 samples, the batch size is defined to 32 and the number of epochs is defined to 2000, then in order to obtain the total number of iterations the following calculations are done:

- Epoch: Number of iterations needed to complete a single dataset train:

$$\boldsymbol{epoch = \lceil \frac{Dataset\ size}{Batch\ size} \rceil = \lceil \frac{1000}{32} \rceil = \lceil 31,25 \rceil = 32} \tag{7.1}$$

- Number of total iterations

$$\boldsymbol{i = epoch \cdot number\ of\ epochs = 64000} \tag{7.2}$$

Several tests in the optimisation stage were performed in order to determine the batch size that will fit optimally the NN and the results were that a batch size equal to the dataset size should be used.

The results are the following:

Table 7.1: Accuracy as a Batch size and numer of epochs function

Batch size	Number of epochs	Result (accuracy)
2410	1000	0.887
512	200	0.864
256	100	0.796
64	28	0.741
32	14	0.668

Note that the total number of iterations remains constant to make a fair comparison.

Generally, this configuration (Batch size = Dataset size) is used with small datasets which is the case. But, the disadvantage of using this batch size is the generalisation capabilities that the algorithm learns. As this generalisation capabilities are required for the application, both batch size equals to dataset size and 512 will be used in following tests.

7.2. Normal classifier (9-Class)

The first test done was the classification of the hole dataset into the 9 different classes (No incident and Incident between 1 to 8).

Below the evaluation metrics of the test can be observed:

Table 7.2: Evaluation metrics for test in 7.2

Evaluation metrics	
Accuracy	0.6747
Precision	0.3954
Recall	0.3916
F1 Score	0.3933

7.3. Normal classifier omitting incident types 1, 5 & 8

Due to poor results in the previous test (9-Class classifier), a second set of tests omitting incidents that theoretically are difficult to detect using accelerometer signals is done. These incidents are Close pass (1), Tailgating (5) and Other (8). In this set of tests, two different parameters change: the batch size and the inclusion/exclusion of user tagged incidents. This user tagged incidents is made in order to demonstrate that user incidents decrease the results of the overall classifier.

7.3.1 Avoiding user tagged incidents

- Batch size: Dataset size

Table 7.3: Confusion matrix in % for test 7.3.1 with batch size equals to dataset size

Confusion Matrix (%)

Predicted									
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	True class
98.62	0.00	0.55	0.00	0.00	0.00	0.00	0.83	0.00	**0**
0.00	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1**
7.69	0.00	**21.15**	15.38	19.23	0.00	0.00	36.54	0.00	**2**
5.80	0.00	7.25	**24.64**	21.74	0.00	0.00	40.58	0.00	**3**
4.17	0.00	5.56	23.61	**27.78**	0.00	4.17	34.72	0.00	**4**
0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	**5**
0.00	0.00	18.18	18.18	0.00	0.00	**9.09**	54.55	0.00	**6**
1.69	0.00	15.25	12.43	14.12	0.00	2.82	**53.67**	0.00	**7**
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**8**

The evaluation metrics and a breakdown by class of the precision and the recall can be observed in the following tables:

Table 7.4: Evaluation metrics and breakdown for table 7.3

Evaluation metrics	
Accuracy	0.6747
Precision	0.3954
Recall	0.3916
F1 Score	0.3933

Class	Precision	Recall
0	0.9624	0.9862
2	0.2157	0.2115
3	0.2576	0.2464
4	0.2857	0.2778
6	0.1111	0.0910
7	0.5398	0.5460

- *Batch size: 512*

Table 7.5: Confusion matrix in % for test 7.3.1 with batch size equals to 512

Predicted

Confusion Matrix (%)

True class

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
100,00	0.00	0,00	0,00	0,00	0.00	0,00	0,00	0.00	**0**
0,00	**0.00**	0,00	0,00	0,00	0.00	0,00	0,00	0.00	**1**
8,70	0.00	**21,74**	17,39	4,35	0.00	17,39	30,43	0.00	**2**
0,00	0.00	11,11	**55,56**	16,67	0.00	0,00	16,67	0.00	**3**
0,00	0.00	0,00	66,67	**0,00**	0.00	0,00	33,33	0.00	**4**
0,00	0.00	0,00	0,00	0,00	**0.00**	0,00	0,00	0.00	**5**
25,00	0.00	50,00	0,00	0,00	0.00	**25,00**	0,00	0.00	**6**
15,00	0.00	0,00	30,00	25,00	0.00	5,00	**25,00**	0.00	**7**
0,00	0.00	0,00	0,00	0,00	0.00	0,00	0,00	**0.00**	**8**

The evaluation metrics and a breakdown by class of the precision and the recall can be observed in the following tables:

Table 7.6: Evaluation metrics and breakdown for table 7.5

Evaluation metrics	
Accuracy	0.6753
Precision	0.3943
Recall	0.3788
F1 Score	0.3707

Class	Precision	Recall
0	0.9326	1.0000
2	0.5556	0.2174
3	0.4167	0.5556
4	0.0000	0.0000
6	0.1667	0.2500
7	0.2941	0.2500

7.3.2 Including user tagged incidents

- Batch size: Dataset size

Table 7.7: Confusion matrix in % for test 7.3.2 with batch size equals to dataset size

Predicted

Confusion Matrix (%)

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
96,53	0.00	0,00	0,69	0,35	0.00	0,00	2,43	0.00	**0**
0,00	**0.00**	0,00	0,00	0,00	0.00	0,00	0,00	0.00	**1**
3,80	0.00	**10,13**	20,25	15,19	0.00	3,80	46,84	0.00	**2**
3,31	0.00	8,29	**37,57**	16,57	0.00	2,21	32,04	0.00	**3**
1,79	0.00	4,46	30,36	**32,14**	0.00	1,79	29,46	0.00	**4**
0,00	0.00	0,00	0,00	0,00	**0.00**	0,00	0,00	0.00	**5**
5,88	0.00	5,88	23,53	17,65	0.00	**11,76**	35,29	0.00	**6**
3,09	0.00	13,06	17,53	19,59	0.00	2,75	**43,99**	0.00	**7**
0,00	0.00	0,00	0,00	0,00	0.00	0,00	0,00	**0.00**	**8**

True class

The evaluation metrics can be observed in table 7.8

Table 7.8: Evaluation metrics for table 7.7

Evaluation metrics	
Accuracy	0.5333
Precision	0.3748
Recall	0.3821
F1 Score	0.3775

- *Batch size: 512*

Table 7.9: Confusion matrix in % for test 7.3.2 with batch size equals to 512

Predicted

Confusion Matrix (%)

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
85,45	0.00	9,09	0,00	0,00	0.00	0,00	5,45	0.00	**0**
0,00	**0.00**	0,00	0,00	0,00	0.00	0,00	0,00	0.00	**1**
5,26	0.00	**15,79**	5,26	31,58	0.00	10,53	31,58	0.00	**2**
11,54	0.00	19,23	**38,46**	0,00	0.00	0,00	30,77	0.00	**3**
6,67	0.00	6,67	13,33	**6,67**	0.00	0,00	66,67	0.00	**4**
0,00	0.00	0,00	0,00	0,00	**0.00**	0,00	0,00	0.00	**5**
14,29	0.00	14,29	14,29	0,00	0.00	**28,57**	28,57	0.00	**6**
6,25	0.00	18,75	12,50	15,63	0.00	18,75	**28,13**	0.00	**7**
0,00	0.00	0,00	0,00	0,00	0.00	0,00	0,00	**0.00**	**8**

True class

The evaluation metrics can be observed in table 7.10.

Table 7.10: Evaluation metrics for table 7.9

Evaluation metrics	
Accuracy	0.4675
Precision	0.3455
Recall	0.3384
F1 Score	0.3376

From the 2 first tests in 7.3.1 and looking at the precision and recall metrics split by incident types in the first test: it can be seen that incident type 0 has good rate of detection, and both precision and recall rates are low for incident types 4 and 6. Moreover, it can be observed that batch size in the first test is not affecting the overall evaluation metrics but in the second test, the batch size equals to the dataset size is performing better than the batch size equals to 512.

From the 2 lasts tests in 7.3.2 including user tagged incidents, it can be clearly seen that using only the no-user-tagged incidents, even if there is less data to train the network, the performance is better.

7.4. Normal classifier with OS separation (avoiding incidents 1, 5, & 8)

This test was made in order to visualise differences between the data coming from different OS, iOS and Android. The goal was to observe if the quality of data recorded by different smartphones was affecting the final classification. In these tests, confusion matrix is omitted due to the lack of information that adds and only the evaluation metrics are represented.

- *Batch size: Dataset size*

7.4 Normal classifier with OS separation (avoiding incidents 1, 5, & 8)

Table 7.11: Evaluation metrics for for test 7.4 with batch size equals to dataset size

iOS

Evaluation metrics	
Accuracy	0.5815
Precision	0.3526
Recall	0.3525
F1 Score	0.3488

Class	Precision	Recall
0	0.9231	0.9600
2	0.3333	0.2105
3	0.3333	0.3077
4	0.0000	0.0000
6	0.0770	0.1250
7	0.4490	0.5116

Android

Evaluation metrics	
Accuracy	0.6684
Precision	0.4105
Recall	0.3920
F1 Score	0.3974

Class	Precision	Recall
0	0.9544	0.9963
2	0.1579	0.2500
3	0.2041	0.1587
4	0.2632	0.2381
6	0.3333	0.2000
7	0.5504	0.5591

- *Batch size: 512*

Table 7.12: Evaluation metrics for for test 7.4 with batch size equals to dataset size

iOS

Evaluation metrics	
Accuracy	0.6688
Precision	0.3898
Recall	0.3721
F1 Score	0.3764

Class	Precision	Recall
0	0.9655	0.9767
2	0.2000	0.2857
3	0.4211	0.4211
4	0.1250	0.1111
6	0.3333	0.2000
7	0.2941	0.2381

Android

Evaluation metrics	
Accuracy	0.6883
Precision	0.4216
Recall	0.4683
F1 Score	0.4643

Class	Precision	Recall
0	0.9222	0.9881
2	0.2308	0.2727
3	0.2400	0.4615
4	0.5000	0.2692
6	0.0000	0.0000
7	0.6364	0.3500

From these 4 tests where the dataset is split depending on the mobile phone used, we can see the importance of the Batch size but also of the amount data. We can see different important improvements and drawbacks depending on where we look at: 1) Due to the amount of data used (less than the first 2 tests, because we split the hole data in iOS data or Android data) it could be noticed a deterioration of all the evaluation metrics, 2) it could be observed that with the batch size of 512 there is an improvement for all the evaluation metrics, in the best cases up to 8% and worse cases up to 1.11%, 3) same happens with the incident types 4 and 6 that have very low precision and recall rates.

7.5. Normal classifier omitting incident types 1, 4, 5 & 8

In this test and as a conclusion from the precision and recall breakdown evaluation metrics results coming from the two first tests, another iteration in the normal classifier is made omitting incident types Close pass (1), Someone approaching head on (4), Tailgating (5) and Other (8).

Table 7.13: Evaluation metrics for test 7.5

- Batch size: Dataset size

Evaluation Metrics	
Accuracy	0.6660
Precision	0.5219
Recall	0.5168
F1 Score	0.5157

- Batch size: 512

Evaluation Metrics	
Accuracy	0.5714
Precision	0.4070
Recall	0.4319
F1 Score	0.4129

7.6. Normal classifier changing window length

In this test, a change in the windowing parameters is done. When the ride is windowed in order to extract the parameters, as explained in 'Adapt Dataset' chapter, a 6 seconds window is set. However, in this test, the window length was set to 3 seconds in order to observe how this change affects the overall performance.

Table 7.14: Evaluation metrics for test 7.6

- Batch size: Dataset size

Evaluation Metrics	
Accuracy	0.5532
Precision	0.4091
Recall	0.4114
F1 Score	0.4081

- Batch size: 512

Evaluation Metrics	
Accuracy	0.5519
Precision	0.3680
Recall	0.3602
F1 Score	0.3532

With this test we can see that using a shorter window is decreasing the overall performance of the classifier from an accuracy of 67% to 55% and from a F1 score of 37% to 35% in the worst case.

7.7. Ternary classifier

This set of tests aimed to obtain a precise view of how each type of incident was detected individually. The goal of the ternary classifier was to detect between no incident, incident and incident type X. There is a total of 8 tests and in order to simplify the results, only the ones with the batch size equal to the dataset size are showed.

In the following tables it can be observed the performance of the algorithm for each of the incident's types. In each type test the confusion matrix as well as the general metrics and a breakdown of the precision and recall for the class 2 is showed. From these results, a clarification has to be done: High values in accuracy displayed in tests done for incident types 5 and 6 are caused by a lack of data of type 2. That produces high values in the evaluation metrics because this table reflects a mean value between all classes. In fact, it can be clearly seen that these 2 incident types are bad detected because of the breakdown in precision and recall. Finally, a summary of the results among all the incident types tests is presented below. In the table it can be observed the precision and recall metrics from the breakdown of class 2 for each of the tests as well as the mean between them.

Table 7.15: Confusion Matrix, Evaluation Metrics and breakdown for class 2 for each of the incident Types

Type 1

C.M. (%)	**Predicted**			True class
	0	**1**	**2**	
	98.24	1.76	0.00	**0**
	1.67	**81.38**	16.95	**1**
	3.45	66.38	**30.17**	**2**

Class	Precision	Recall
2	0.3017	0.3017

Evaluation Metrics	
Accuracy	0.8302
Precision	0.6987
Recall	0.6993
F1 Score	0.6990

Type 2

	Predicted			
C.M. (%)	**0**	**1**	**2**	True class
	99,05	0,95	0,00	**0**
	4,11	**92,15**	3,74	**1**
	3,28	83,61	**13,11**	**2**

Class	Precision	Recall
2	0.2857	0.1311

Evaluation Metrics	
Accuracy	0.9107
Precision	0.7132
Recall	0.6810
F1 Score	0.6874

Type 3

	Predicted			
C.M. (%)	**0**	**1**	**2**	True class
	98,99	0,80	0,20	**0**
	1,88	**86,68**	11,44	**1**
	5,97	76,12	**17,91**	**2**

Class	Precision	Recall
2	0.1622	0.1791

Evaluation Metrics	
Accuracy	0.8806
Precision	0.6760
Recall	0.6786
F1 Score	0.6771

Type 4

	Predicted			
C.M. (%)	**0**	**1**	**2**	True class
	99,02	0,78	0,20	**0**
	2,64	**90,00**	7,36	**1**
	3,17	88,89	**7,94**	**2**

Class	Precision	Recall
2	0.1111	0.0794

Evaluation Metrics	
Accuracy	0.8950
Precision	0.6563
Recall	0.6565
F1 Score	0.6555

Type 5

	Predicted			
	0	**1**	**2**	True class
C.M. (%)	**98,35**	1,65	0,00	**0**
	2,36	**96,28**	1,35	**1**
	0,00	87,50	**12,50**	**2**

Class	Precision	Recall
2	0.2000	0.1250

Evaluation Metrics	
Accuracy	0.9610
Precision	0.7120
Recall	0.6905
F1 Score	0.6983

Type 6

	Predicted			
	0	**1**	**2**	True class
C.M. (%)	**98,38**	1,62	0,00	**0**
	3,21	**96,28**	0,51	**1**
	15,38	84,62	**0,00**	**2**

Class	Precision	Recall
2	0.0000	0.0000

Evaluation Metrics	
Accuracy	0.9621
Precision	0.6430
Recall	0.6489
F1 Score	0.6459

Type 7

	Predicted			
	0	**1**	**2**	True class
C.M. (%)	**97,82**	1,45	0,73	**0**
	2,33	**73,43**	24,24	**1**
	2,40	67,66	**29,94**	**2**

Class	Precision	Recall
2	0.3185	0.2994

Evaluation Metrics	
Accuracy	0.7621
Precision	0.6703
Recall	0.6706
F1 Score	0.6703

Type 8

C.M. (%)	**Predicted**			True class
	0	**1**	**2**	
	99,80	0,20	0,00	**0**
	5,13	**84,19**	10,68	**1**
	3,03	75,76	**21,21**	**2**

Class	Precision	Recall
2	0.2877	0.2121

Evaluation Metrics	
Accuracy	0.8547
Precision	0.6923
Recall	0.6840
F1 Score	0.6860

Table 7.16: Summary table: precision and recall breakdown for each of the 8 types tests

Type test	1	2	3	4	5	6	7	8
Precision class 2	0.3017	0.2857	0.1622	0.1111	0.2	0	0.3185	0.2877
Recall class 2	0,3017	0,1311	0,1791	0,0794	0,125	0	0,2994	0,2121
Mean class 2	0,3017	0,2084	0,17065	0,09525	0,1625	0	0,30895	0,2499

7.8. Binary classifier

The final test was to build a binary classifier in order to classify between incident and no incident. This classifier is useful in order to improve the incident detection algorithm that is already implemented in SimRa app.

- Batch size: dataset size

Table 7.17: Confusion Matrix and Evaluation Metrics for test in 7.8 and for batch size equal to dataset size

C.M. (%)	**Predicted**		True class
	0	**1**	
	98.82	1.18	**0**
	4.00	**96.00**	**1**

Evaluation Metrics	
Accuracy	0.9743
Precision	0.9875
Recall	0.9600
F1 Score	0.9735

- *Batch size: 512*

Table 7.18: Confusion Matrix and Evaluation Metrics for test in 7.8 and for batch size equal to 512

C.M. (%)	**Predicted**		**True class**
	0	**1**	
	98.82	1.18	**0**
	5.80	**94.20**	**1**

Evaluation Metrics	
Accuracy	0.9675
Precision	0.9848
Recall	0.9420
F1 Score	0.9630

When using a binary classifier, the algorithm performs pretty good. And results using the batch size equals to the dataset are slightly better which shows us that trying to perform basic classifications there is no need to use lower batch sizes that impact in the convergence of the gradient that provokes a diminution of the overall performance.

Chapter 8

Integration with SimRa App

In this order to integrate the developed classifier, 2 programs have to be made. These two programs are needed in order to preprocess the ride data coming from SimRa App and to post process the data once the neural network has classified all the ride windows.

- *Pre process - AdaptRide*

The pre process stage take place once the ride is finished and the file with the accelerometer recorded data is generated. Afterwards, the AdaptRide program, available in a GitHub Repository [22], take this data and outputs a CSV file with the data that will be used by the trained Neural Network to find the incidents that may have been occurred during the ride. The main flow of AdaptRide is the same as the AdaptDatabase but some minor differences are described below:

- A new feature to avoid the first and last seconds of the ride is available. This is added in order to omit possible false positives that may occur during the start and end of the ride.
- The output incident type associate to each of the windows is set as 0 for all of the windows. That is because the program doesn't know which is the associate incident type for any of the windows due to the fact that the incidents must be found by the neural network.

- *Post process: SimRaNN Test*

The post process stage take place once the AdaptRide program has generated the output CSV file that contains all the ride data in a good format to use it as input of the neural network. Afterwards, the SimRa Test program, available in a GitHub Repository [23], takes this data and classify each of the windows of the ride as incident or not. This is made using a pre trained neural network. The output of this program will be the window numbers that has been estimated by the neural network as probable incidents.

This program has some tune parameters that are described below:

- Filter: The neural network associate to each of the ride windows a probability value. With this parameter the threshold probability to consider an incident is set.
- MaxIncidents: This parameter is used to fix a maximum number of positives that the neural network will output.

Chapter 9

Evaluation of the elapsed times

In order to assess the viability of the overall procedure and classifying algorithm as well as integrate this project in the smartphone app, some tests regarding processing times have been performed. All this test has been performed under the following hardware devices:

- Processor: 3,1 GHz Dual-Core Intel Core i7
- RAM Memory: 16 GB 1867 MHz DDR3

9.1. Optimizing Network Hyperparameters

The first one is regarding the optimization procedure. Although this stage is an optional procedure that only tries to find better parameters when defining the architecture of the Neural Network, an analysis of the elapsed time is done. The test is done under the following characteristics:

- Iterations per candidate: 1.000
- Evaluated Candidates: 10
- Dataset size: 2.521 entries

The total elapsed time: 0h 46min 32sec 202ms

Note: A proposal for this procedure is to run it in a cloud server and upgrade the number of evaluate candidates to search for a possible better parameter, as the final output generated for the program is a folder with 4 files containing the final score obtained, the model configuration (in JSON), the iterations done and the model itself (in a binary file).

9.2. Training Network

The second test is regarding the training of the Neural Network. As well as the previous test, this stage is a procedure that has to be run only once and it can be run in a computer instead of in a smartphone. Unlike the first procedure, this one has to be run mandatorily. The test is done using the code in [21] under the following characteristics:

- Batch size: 512
- Iterations to the dataset: 5.000
- Dataset size: 3.352 entries

The total elapsed time: 38min 34sec 145ms

9.3. Ride evaluation and incident classification

The final test is regarding the ride windowing and classification using the neural network. This procedure is the one that must be done in the app when the ride ends because will be the SimRa App the one who will perform these operations. The test is done with a ride that lasts for around 25 min. The duration of the two procedures separately are shown below:

- Windowing ride: 655 ms
- Classifying the windowed ride: 2 sec 646ms

Chapter 10

Future considerations

Once ended this project, some considerations have shown up. These different tests could be interesting in order to evaluate a possible improvement in the actual incident detection algorithm and from this point more analysis can be done.

- *Searching the incident in a near position when the user manually tagged it.*

In order to properly use the user tagged incidents and don't decrease the overall performance of the classifier, the following approach could be tried: Assuming that user tagged incidents deteriorate the performance because they are not precisely located, it can be concluded that the signals that define the incident that the user has reported are in fact signals coming from before or after the real incident instant. In order to avoid this misplacement of the incident, an algorithm based on the binary classifier could be developed in order to search the real incident signals in a time frame around the marker of the user. Once this search is done, the incident marker can be set in a more precise location and its usage could improve the training of the algorithm instead of deteriorate it.

- *Splitting incident number 7*

When we try to classify incident number 7, which corresponds to 'Dodging an obstacle' a sub classification could be done. The reason is that a distinction between two different types of dodging could be done: fast dodging and slow dodging. A fast dodging could be in fact an emergency maneuver because someone crossed our way. A slow dodging could be for instance going around a bus stop. These two types of dodging could have very different statistics and signals. Both are interesting because a fast dodging is important regarding the detection of near miss incidents and slow dodging is important when planning a bikeway. A variation in the window length used to window the ride, once this is finished, could influence the detection of one or the other. Depending the final scope of the Neural network the window could be adapt to detect one type or the other.

List of Figures

List of Tables

Bibliography

[1] Einstein Center Digital Future. Simra project. https://www.digital-future.berlin/forschung/projekte/simra/.

[2] Einstein Center Digital Future. Simra dataset. https://github.com/simra-project/dataset.

[3] Albert Sanchez Fuster. Extractdata github repository. https://github.com/albertsanchezf/ExtractData.

[4] Albert Sanchez Fuster. Accelerometer data enhancement github repository. https://github.com/albertsanchezf/AccelerometerDataEnhancement.

[5] Przemek Maziewski, Adam Kupryjanow, K. Kaszuba, and A. Czyzewski. Accelerometer signal pre-processing influence on human activity recognition. pages 95 – 99, 10 2009.

[6] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411 – 430, 2000.

[7] Matlab. Pca and ica matlab package. https://www.mathworks.com/matlabcentral/fileexchange/38300-pca-and-ica-package.

[8] A. S. Abdull Sukor, A. Zakaria, and N. Abdul Rahim. Activity recognition using accelerometer sensor and machine learning classifiers. In *2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA)*, pages 233–238. IEEE, mar 2018.

[9] Davide Figo, Pedro Diniz, Diogo Ferreira, and João Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14:645–662, 10 2010.

[10] Albert Sanchez Fuster. Adapt database github repository. `https://github.com/albertsanchezf/AdaptDatabase/`.

[11] Ivan Pires, Nuno Garcia, Nuno Pombo, Francisco Flórez-Revuelta, Susanna Spinsante, Maria Teixeira, and Eftim Zdravevski. Pattern recognition techniques for the identification of activities of daily living using mobile device accelerometer, 09 2018.

[12] Abdulmajid Murad and Jae-Young Pyun. Deep recurrent neural networks for human activity recognition. *Sensors*, 17:2556, 11 2017.

[13] S. L. Lau and K. David. Movement recognition using the accelerometer in smartphones. In *2010 Future Network Mobile Summit*, pages 1–9, June 2010.

[14] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. Physical Human Activity Recognition Using Wearable Sensors. *Sensors*, 15(12):31314–31338, dec 2015.

[15] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235 – 244, 2016.

[16] Yusuke Kobana, Junji Takahashi, Naofumi Kitsunezaki, Yoshito Tobe, and Guillaume Lopez. Detection of road damage using signals of smartphone-embedded accelerometer while cycling. 09 2014.

[17] S Candefjord, L Sandsjö, R Andersson, N Carlborg, A Szakal, J Westlund, and B A Sjöqvist. Using Smartphones to Monitor Cycling and Automatically Detect Accidents-Towards eCall Functionality for Cyclists. *Proceedings, International Cycling Safety Conference, Göteborg, Sweden, November 18-19*, (November), 2014.

[18] DL4J. Deep learning 4 java. `https://deeplearning4j.org/`.

[19] Albert Sanchez Fuster. Simra nn optimization github repository. https://github.com/albertsanchezf/SimRaNN_Optimization/.

[20] Albert Sanchez Fuster. Simra nn binary github repository. https://github.com/albertsanchezf/SimRaNN_Training/.

[21] Albert Sanchez Fuster. Simra nn binary github repository. https://github.com/albertsanchezf/SimRaNN_Binary/.

[22] Albert Sanchez Fuster. Adapt ride github repository. https://github.com/albertsanchezf/AdaptRide/.

[23] Albert Sanchez Fuster. Simra nn test github repository. https://github.com/albertsanchezf/SimRaNN_Test/.