POLITECNICO DI TORINO
EURECOM

Real-time automatic transport mode detection on smartphones using Recurrent Neural Networks

Le réseau neuronal récurrent dans la détection en temps réel du mode de transport sur smartphone

by

Arnaudo Valerio

Supervised by:
Pronello Cristina (Industrial Supervisor - UTC Compiègne)
Härri Jérôme (Academic Supervisor - Eurecom)
Malnati Giovanni (Industrial Supervisor - Politecnico di Torino)

in

Ict for Smart Societies
Double degree exchange in Data Science and Engineering

March 2018

“Tout est possible quand tu sais qui tu es
Tout est possible quand t’assumes qui tu es”

Soprano feat. Marina Kaye - Mon Everest

POLITECNICO DI TORINO
EURECOM

Abstract

Ict for Smart Societies
Double degree exchange in Data Science and Engineering

Master Thesis

by **Arnaudo Valerio**

The design of an efficient public transport system depends on the users' needs and their travel patterns and, since mobility change over time, collecting up-to-date data has become essential for local administrations. To this end, a smartphone application has been developed to understand users' travel behaviour. Nowadays, smartphones are an integral part of people's lives and their potential has been exploited to support people in daily activities, such as calls, chat, entertainment as well as mobility. The thesis work aims at improving the mode detection of the above application, identifying the transport mode (walk, bike, car, etc.) used by the traveller. In order to perform the mode detection, an algorithm using Recurrent Neural Network has been designed and tested in real-life scenarios. Battery life conservation and real-time results have been taken into account to improve the user experience. Compared to Google APIs, the algorithm is able to better recognize the diverse motorized modes and with higher accuracy.

POLITECNICO DI TORINO
EURECOM

Résumé

TIC pour les sociéts intelligentes
Double diplôme en science et génie des données

Thèse de Master

par **Arnaudo Valerio**

Afin de concevoir efficacement un système de transport public, il faut prendre en considération les besoins des utilisateurs et leurs habitudes de déplacement et, étant donné que ces informations évoluent dans le temps, les mettre à jour est devenu essentiel pour les administrations locales. De ce fait, une application mobile nommée *Mobilité Dynamique* a été développée afin de comprendre les différents besoins de mobilité des utilisateurs. De nos jours, les smartphones font une partie intégrante de la vie courante et leurs utilisateurs dépendent sur eux pour effectuer des appels, envoyer des messages, tre informé des dernières actualités, se divertir, avoir des informations sur le trafic et même trouver leurs chemins. Cette thèse vise à améliorer les performance de l'application *Mobilité Dynamique* en identifiant le mode de transport (marche, vélo, voiture, etc.) du voyageur. Afin de réaliser la détection de mode, un algorithme utilisant Réseau de Neurones Récurrents a été conçu et testé dans des scénarios réels. La prolongation de la durée de vie de la batterie et la préservation de cette dernière ainsi que le besoin d'avoir le mode de transport en tant réel ont été pris en compte pour améliorer l'expérience de l'utilisateur et l'inciter à utiliser l'application. Comparé aux API Google, l'algorithme est capable de reconnaître les différents modes de transport motorisés avec une plus grande précision.

Acknowledgements

Grazie/merci/thanks:

Ai Professori Pronello e Malnati per il sostegno e la professionalità dedicatami nel raggiungimento di questo traguardo
Au professeur Härri pour la disponibilité d'aide en cas de difficulté
Ai miei genitori, base di ogni mio sogno e successo senza la quale non si sarebbero mai avverati
A mia Sorella, amica e faro della mia vita su cui posso sempre contare
Ai miei nonni, sostento di amore e energia per non mollare mai davanti alle difficoltà
A Nesrine, pezzo mancante del puzzle che ha reso la mia vita completa Nhebek
Ai miei zii e cugini, sostegno silenzioso ma fondamentale nella mia riuscita
Enrico, Santi, Alain, Fabio, Michele, Corrado e Matteo amici indispensabili
Manuela, Francesca e Chiara amiche sempre presenti
Au département GSU pour le séjour confortable à l'UTC de Compiégne
A tutti i miei amici che mi hanno sempre sostenuto
A mes cousins et amis français
To all my friends all around the world
A chi c'era, chi c'è e chi ci sarà

Contents

List of Figures

List of Tables

Abbreviations

API	**A**pplication **P**rogramming **I**nterface
CPU	**C**entral **P**rocessing **U**nit
FFT	**F**ast **F**ourier **T**ransform
GIS	**G**eographics **I**nformation **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
GSM	**G**lobal **S**ystem for **M**obile Communications
GTFS	**G**eneral **T**ransit **F**eed **S**pecification
HMM	**H**idden **M**arkov **M**odel
LSTM	**L**ong **S**hort **T**erm **M**emory
MLP	**M**ulti **L**ayer **P**erceptron
NN	**N**earest **N**eighbour
OTP	**O**pen **T**rip **P**lanner
RNN	**R**ecurrent **N**eural **N**etwork
SVM	**S**upport **V**ector **M**achines

A Nonna Franca

Chapter 1

Introduction

The transport planning aims at defining the most suitable facilities and services to achieve the highest customer satisfaction since every person needs to travel from one location to another one for multiple reasons (work, school, shopping, leisure, etc.). Designing the transport facilities depending on people's needs is very crucial, thus transport planning is the core of a functional territory. Indeed, a badly conceived transport system is perceived as inefficient and, therefore, will never be used with high social and economic impacts.

The most important information to carry out an accurate and effective transport plan is people's travel behaviour or mobility pattern. The simplest way to retrieve this information is to carry out a survey and ask people questions about their travel habits. This method has, however, a lot of weaknesses. In fact, the collection of data from a representative sample is expensive and time consuming; in addition, following the continuous evolution of mobility patterns requires periodical surveys that cannot be afforded by the local administrations. The recent approaches using web-questionnaire to collect data are less expensive but not everyone is willing to spend time to answer to a survey; the consequence is that samples are quite biased and the efforts to balance the sample increase the costs of the survey. Another problem related to the reduction of both survey costs and the fatigue of the respondents is the trade-off between detailed information (long survey) and response time. Collecting all daily travel routines of a person for more than one day through a survey is not affordable and get a large sample is quite challenging.

To overcome all the problems stated above, a smartphone application, *Mobilité Dynamique* (*Dynamic Mobility* in English), has been developed. Exploiting the potential

of today's smartphones, the application aims at understanding the users' travel patterns.

This innovative method has advantages for both the local administration/transport companies and users. In fact, the above organizations will save many resources and, most importantly, they will have the availability of continuous updated data. On the other hand, users will be more aware of their travel patterns because the application suggests to them ways to improve their mobility, considering the time, the cost and the environmental footprint of their travels. The users become fully aware about the time spent any day travelling using the different modes, understand which the related cost is as well as how many pollutant emissions they produce according to the chosen mode. The comparison with potential alternatives can help people to increase their awareness and evaluate the best option, considering the diverse personal and collective effects of their mobility.

The richness of the above information comes from the application's ability to recognize the transport mean used by the traveller. Thus, the mode detection is one of the most important information we need in order to carry out more accurate and efficient transport plan because it allows understanding people travel behaviour. To this end, such information needs to be extremely accurate.
The thesis objective is to find the most suitable way to understand the transport mean used by travellers, named *Transport Mode Detection*.

The thesis is articulated in four chapters. The first chapter 2 deals with the state-of-the-art literature, showing the different approaches developed by researchers all over the world to perform the mode detection. The second chapter 3 presents the objectives of the thesis work and the methodology developed to reach the ambitious objectives. The results are described in the third chapter 4 and the conclusive remarks are given in the fourth and last chapter 5.

Chapter 2

State-of-the-art literature

The transport mode detection is a topic that has been the centre of many research projects in the past decades. The continuous evolution of technology, specifically the smartphones, has a major role in the research progresses. Nowadays, the number of smartphone users worldwide overpasses the two billions Newzoo (2017). Researchers followed different ways to achieve the highest accuracy in the mode's assignment. In this chapter, we rely on two different characteristics to compare the various research papers covering this topic:

- the methodology used to perform the mode's assignments.
- the sensors used to perform the study.

2.1 Mode detection methodology

The algorithms analysed in the following sections belong to the group of classification algorithms. This category of algorithms aims at assigning a specific class based on the characteristic of the input:

$$Y = f(X)$$

The classification algorithm f assign to the class Y the input X. The number and the type of different classes are predefined.
As mentioned above, several methodologies have been investigated to perform the mode detection. The classification methods can be divided in two categories: fuzzy rules and machine learning.

2.1.1 Fuzzy rules

The fuzzy rules' algorithms are the simplest method to perform classification. A predefined number of questions are formulated based on the input's type. The researcher decides the questions and, therefore, no specific knowledge on information technology is required to code the classifier. The input query the questions and, based on the answers, a class is assigned. Three research projects have tried to perform mode detection using fuzzy rules.
The work made by Sauerlnder-Biebl et al. (2017) provides three steps:

- in the first step, the data position of a tracked person is divided into smaller segments;
- in the second steps the mode detection is performed using fuzzy rules (maximum speed, mean speed, start acceleration, change of heading after stop, etc.). Those variables are calculated using only the GPS position with a regular rate of at least one position every 4 seconds. The detected modes are walk, bicycle, bus, car and train;
- the last step consists of further logical checks about the detected mode.

Another example of using fuzzy rules is investigated by Schüssler and Axhausen (2009). GPS points are used as data inputs and a post-processing procedure is implemented (no real-time). After a cleaning and smoothing stage where some data are removed, a trip and activity detection are performed. This step tries to recognize if the user have stopped the trip for some reasons and, in this case, it labels this event as a new activity. Finally, the mode detection is performed using the sequences which were recognized as a trip. The fuzzy features are: the median of the speed distribution and the ninety-fifth percentiles of the speed and acceleration distributions. The detected modes are walk, bicycle, car, public transport and train.

The last paper using fuzzy rules is the work done by Shin et al. (2015). Both the accelerometer and the position are used with a sampling rate of one per second. The position is retrieved by the network estimate (including network use, default Android data synchronizing service enabled, 3G network service, Wi-Fi, a messenger application, a Google email application, and default media sync functions). The detection process first consists of detecting the walking segments and using them to partition the overall activities. Subsequently, if the speed is greater than 7 km/h, the travel mode detection is performed by estimating the acceleration profile and comparing it to the one that is

calibrated by the authors. The detected modes are walk, trains, bus and car.
The above researches obtained good results in detecting walking and biking but they had some difficulties in detecting the motorized means. Since the method uses basics rules, it is difficult properly setting the selected features. More and more the computer processing power increases, so it is more convenient to let the algorithms find the proper features' settings.

2.1.2 Machine learning methods

Machine learning methods are those type of algorithms able to perform tasks without being explicitly programmed. The algorithm improve the performances by learning from the data. Different approaches are present in literature. The following sections investigates the methods listed below:

- Bayesian methods
- Decision tree / Random forest
- Random Subspace method
- Hidden Markov Model
- Support vector machines
- Nearest neighbor
- Neural networks

2.1.2.1 Bayesian methods

The Bayesian classifier is a statistical method that performs classification based on the Bayes' theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

The results can be expressed as the probability of assigning the class Y observing the input X and it is called the posterior probability. The probability of X belonging to each class is calculated and then the class with the highest probability is selected and assigned to X. The term P(Y) is called the prior probability and it depends on the distribution of the training data on the respective class (very often represented by a uniform distribution). The most important term in the formula is P(X—Y) called likelihood that is

the probability to assign X given the class Y. The difficulty is to find the features that separates the classes in such a way that we have the lowest misclassification occurrences.

The Geurs et al. (2015) work applies the Bayesian approach based on the following statistics: speed patterns, sensor data characteristics, infrastructure network and personal trip history. They designed an application to collect one GPS record every two seconds once the application has detected the movement (using GPS, accelerometer, Wi-Fi and cellular-ID information). When a trip ends, the data are uploaded to a back-end server, processed and finally classified in one of the following modes: walk, bicycle, trains, bus and car. Finally, the user is asked to complete a recall survey on the correctness of the assigned mode.

The GoEco! application (Bucher et al., 2016) is one of the most complete suite for transport mode detection available in both the iOS and Android stores. Unfortunately, the application is no longer operational since the research project ended. The main objective was to reduce the environmental impact and suggest everyday solutions to increase sustainable mobility. The data collection is performed by another application, named Moves. This fitness application is able to recognize Walk, Run and Bicycle while all other transport modes are classified as Transport. Through the Moves APIs, the GoEco! application queries the Open Trip Planner (OTP) to identify possible connections using the available transport systems. Moreover, thanks to the results from the OTP, a list of 16 statistics are defined to instruct the Bayesian classifier and, more specifically, the Nave one where all the features are considered as non-correlated. The classifier is able to recognize foot (including both walking and running), bike, electric bike, kick scooter, car, electric car, motorbike, scooter, bus, train, tram, plane and ship. Finally, the user is requested to confirm the trip.

The work done by Montoya et al. (2015) is divided in two phases. The first phase, named filtering, tries to derive the most likely itinerary from a transport network for the modes walk, road vehicles, bike and rail. They use a probabilistic algorithm based on a dynamic Bayesian network. The second phase, named smoothing, infer the exact transport mode of road vehicles i.e. car and bus and the transport mode of rail i.e. metro and tram. To this end, the researchers determine the possible matching with GTFS (General Transit Feed Specification) routes. All the details of the methodology and the steps of the research are explained in Pluto (2014). The used sensors are the GPS (1 Hz rate), Wi-Fi and Cellular network for the location observations. For the dynamic observations, an accelerometer with a sampling rate of 20 Hz is used. The

Bluetooth, Wi-Fi and cellular network are used to infer the user's environment.

All the above works got good results on the walk and bike modes. Geurs et al. (2015) had problem to detect the motorize modes (notably the bus) for short trips, while the car and trains accuracy improve with longer trips. The GoEco! application (Bucher et al., 2016) got overall good accuracy but it was not well performant concerning car, bus and train. Instead Montoya et al. (2015) got very impressive results in bus detection, but also the train, tram and car are good enough.

The above researches show how GTFS can makes the difference for the results correctness. The only disadvantage is that GTFS are limited to the analysed cities and not valid everywhere. About the Bucher et al. (2016) work, is interesting the decision of using an existing application, already optimized, to record the data (even if sometimes the application lacks some important GPS points). They also pretend to recognize lot of different modes that made the detection even more difficult. To make Bayesian method working, the features' choice is very important. The features should have a likelihood clearly separated among all the modes, thus allowing very high accuracy.

2.1.2.2 Decision tree and Random Forest classifier

Decision tree is a powerful mechanism to analyse large amounts of data. The goal is to build a tree based structure where the nodes represent the questions used to classify the input. It is based on an idea similar to the Fuzzy rules' approach, but the questions are automatically prepared by the algorithm. The questions are chosen in a way that maximizes the distinction between classes and are based on the features of the input data. The figure 2.1 shows a decision tree schema.

Random forests is a method that classifies the input data based on multiple decision trees (ensemble learning) that merge their results into one. The several classifiers (trees) are created by randomly sampling the training data. This approach is an example of the application of the bootstrap aggregation methods (to merge different classifiers' results) and its major advantage is the reduction of the over-fitting [1] problem of single decision tree. However, its major disadvantages are the higher computational and time complexity.
The proposed work by Manzoni et al. (2011) uses the accelerometer with a rate of 25 Hz to perform mode detection. Moreover, GPS data at a rate of 1 Hz and a wireless internet

[1] The tendency of a machine learning algorithm of being too much precise on a specific data set and not able to generalize to all the data.

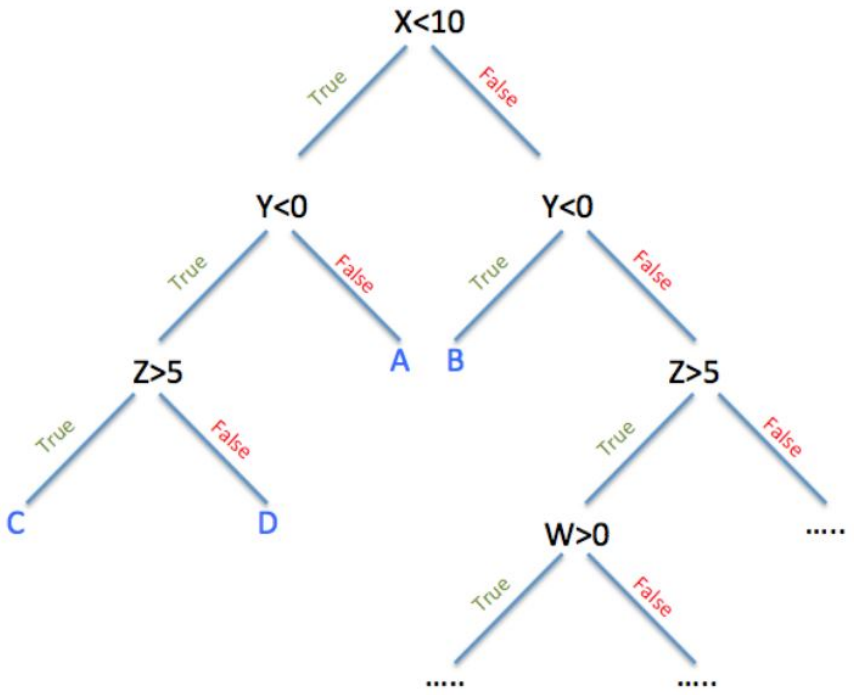

FIGURE 2.1: decision tree schema

connection is optionally required for the computation of the travelled distance. After a pre-processing step, the magnitude of the acceleration is estimated. Subsequently 32 FFT [2] coefficients are computed and used as input features to the decision tree algorithm. Another input feature is the signal variance computed as the sum of the FFT coefficients. The detected modes are Still, Walk, Bike, Car, Motorcycle, Train, Bus and Metro.

Montini et al. (2015) work uses random forest classifier. fuzzy rules are also used but subsequently replaced. The sensors used by the smartphones are GPS with a rate of 1Hz, Wi-Fi for location (helping the GPS) and accelerometer with a rate of 5Hz. A travel diary generation is performed in three steps. First the raw data is filtered to drop non accurate measurements. Second, activities and trips are mainly identified based on point clouds, signal gaps and changes in the accelerometer signal. Finally, the mode is detected by the random forest classifier using the speed, accelerometer variables and also the corrections made by the users. Unfortunately, the technical part was not too much detailed. The detected modes are Walk, Bike, Car, Bus, Rail and Metro. Battery drain is depicted as a huge problem.

A Random Forest classifier was employed by Wang et al. (2017). Only the GPS with a rate of one record per second was used. Some particular rules are applied to recognize the subway trips. These rules were applied because in the subway the GPS signal is not reliable (at least for satellites in view). These rules rely on a small number of reliable

[2] Fast Fourier Transform.

GPS points to determine when the subject enters or leaves the subway. The random forest classifier is employed using 22 different features that can be divided in four main categories which are Speed, Acceleration, Orientation and Distance/Duration. The different steps of the random forest classifier and the mentioned attributes are detailed in the paper. The detected modes are Walk, Bike, e-Bike, Car, Bus and Subway.
Xiao et al. (2012) presentes a decision tree approach which uses the GPS and GSM along with the accelerometer data to perform mode detection. The sensors' sampling rate however is not specified. After a pre-processing step, traces of the positions are generated using GPS. The traces are also identified by detecting the stops which are identified as the absence of movements within a fixed amount of time. The detected modes are limited to Bus, Mass rapid transit, Taxi and Running. The features that have been used to train the decision tree are the standard deviation of the magnitude of force, the maximum moving speed and the average moving speed. Unfortunately, the work did not present any results.

As mentioned above Xiao et al. (2012) did not present any results. Montini et al. (2015) presented some results but through a plot and it is not detailed which mode were correctly detected or not. There is only the accuracy related to single users. The others two works got good results in detecting Walk and Bike. Wang et al. (2017) got some problems in detecting the Bus and the e-Bike, on the other hand the others modes got over 80% of accuracy. Finally, the work done by Manzoni et al. (2011) got very promising results for motorcycle, car and bus with over 80% accuracy. Instead the metro and still had some misclassification. For example the mode still had been classified as car for more than 10% of cases.
The decision tree approach is very promising. It is compared also with others methods in the next mode detections approaches. The capability of merging more decision trees in one classifier, with random forest, let the approach go one step further. Both they need an accurate selection of features because they are essentials in the creations of trees.

2.1.2.3 Random Subspace Method

The random subspace method belongs to those type of algorithms that try to merge different classifiers into one. This approach is an example of applying the bootstrap aggregating or bagging methods (like the random forest algorithms). The peculiarity of this method is that the features used to train the classifiers are also randomly sampled. Nitsche et al. (2012) applies the random subspace method but it is not specified which type of classifier has been used. It only states that statistical classifiers are employed. An accelerometer with a rate of 100 Hz and the GPS with a rate of 1 Hz are used.

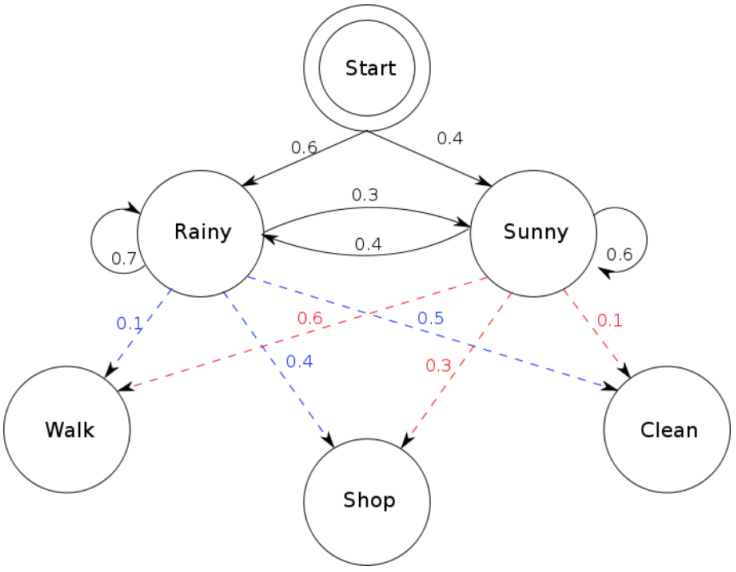

FIGURE 2.2: Hidden Markov Model example

The extracted features are numerous. In fact, seven features are extracted from the GPS, 64 features are extracted from the accelerometer and one features represents the detected motion. The complete list can be found in the paper. Considering the high features' dimensionality in comparison to the small size of the training data the authors decided to use this method. The detected modes are Still, Walk, Bike, Motorcycle, Car, Bus, Electric tramway, Metro and Train. The results are very good only for the modes walk and bike. For all the others modes the accuracy is lower than 80% with lot of misclassification for the modes tramway, metro and train. Only few works used this method (Widhalm et al., 2012) explained next chapter. The results are not very promising even lots of features are used. High dimensionality, if is not supported by large datasets, is not convenient as demonstrated by these results. For mode detection the random subspace method is not appropriate.

2.1.2.4 Hidden Markov Model

Markov model is a stochastic model used to represent the changing states of a system. The transition between the states depends on defined probabilities that are not affected by the previous transitions and are independent for each state. The transitions between one state and another are defined a priori. Though the Hidden Markov Model has the same ideology, the states and the respective transitions are hidden. The outputs of the states that correspond to the final classes are visible. In the figure 2.2, the weather's conditions are the hidden states and the activities are the outputs of the hidden states.

Widhalm et al. (2012) suggests a method composed by two steps. The first step trains an ensemble of 100 different classifiers on randomly selected subspaces of the feature space. The ensemble method is the random subspace method explained in the previous chapter where the base learners are the decision trees. The averaged posterior class probabilities from the previous step are used as input to the discrete Hidden Markov Model. The hidden states correspond to the classification categories i.e. the mode detected. The used sensors are the accelerometer with a rate of 50 Hz and the GPS with a rate of 1 Hz which was used along with the cellular network to determine the location. A set of 77 different features is extracted from the collected data. The categories of features can be grouped in different statistics: velocity, acceleration, deceleration, angular velocity, standard deviation and the power spectrum. The detected modes are Walk, Bike, Car, Bus, Subway, Train, Motorcycle and Tram.
The work done by Reddy et al. (2008) is divided in two steps. In fact, a decision tree is followed by a Hidden Markov Model. The approach is not provided with as much details as the one of Widhalm et al. (2012). The employed sensors are the GPS with a rate of 1 Hz and the accelerometer with a rate of 35 Hz. The period of classification is one second with an overlap of 0.5 seconds. The extracted features are the variance, energy, the sum of FFT coefficients from the accelerometer and the speed from the GPS. The detected modes are Still, Walk, Run, Bike and Motor (no subcategories).

The two works just analysed present different results. The work done by Widhalm et al. (2012) has impressive results for bike and good results for walk, bus and motorcycle. The others mode got results under 72% of accuracy. As explained in the previous chapter the random subspace method gives no good results. Instead the work done by Reddy et al. (2008) got very impressive results for all the modes and for all the classifier they tested. The only problem is given by the bike for the naive Bayes classifier. All the others results got in average 90% of accuracy. The best classifier is the one they choose: decision tree followed by a discrete Hidden Markov Model. It is important to notice that the dataset is very little and the classifier are trained singularly per each user and subsequently tested on the respective user.
Few works used Hidden Markov Model as mode detection. The difficulty stays in the definition of the hidden states and their respective training. Standing on the work proposed by Reddy et al. (2008),this approach can give very promising results.

2.1.2.5 Nearest Neighbour

The nearest neighbour approach is one of the simplest methods to perform classification. Based on the features' data, the new input search the K (arbitrary positive integer greater than zero) closest training data and counts which is the most representative

class i.e. the most similar class based on the features. The figure below shows how the algorithm work. In the figure 2.3, image A, the assigned class will be a minus, in the second image B the class will be unknown and the third image C the assigned class will be a plus.

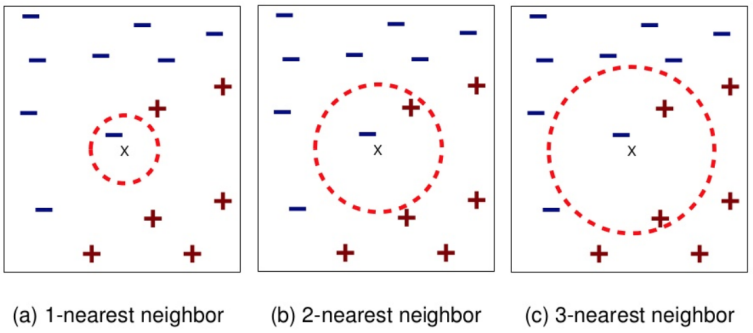

FIGURE 2.3: nearest neighbour example

Sonderen (2016)'s work uses the accelerometer data with a sampling rate of roughly 140 Hz. Since the rate is not perfectly timed, a subsampling is performed (reduced rate as pre-processing) and different rates are tested. Moreover, different time frames are tested and the time periods are 1, 5, 10, 20 and 30 seconds. The used features are the average value, the median value, the minimum value, the maximum value and the interquartile range. The expectation is that the nearest neighbour will have the highest accuracy in comparison to the decision tree and the random forest approaches that were also tested. The detected modes are Walk, Bike, Run and driving a Car.
Martin et al. (2017) also tried different approaches to perform mode detection such as the nearest neighbour, the Movelet approach and the random forest. The Movelet approach, in the literature, is usually applied to predict human movements and not to perform mode detection. It consists of a machine learning technique based on matching time series sequences. The used sensors are the GPS with a rate of one sample per second and the accelerometer with a rate of 5 Hz. From the GPS data, a vector of speed values is created. The Movelet method creates partitions of both the accelerometer and speed time series data into segments (Movelets) then clusters the segments that are known to be from the same mode. The nearest neighbour and the random forest approaches use as features the mean, median, variance, minimum, maximum, interquartile range, 20th percentile and 80th percentile of the acceleration and the speed data. The detected modes are Walk, Bike, Car, Bus and Rail.

The two works present different results. Sonderen (2016) claims that nearest neighbour is better than decision tree and random forest approaches. Instead Martin et al. (2017) state the opposite. In both works the accuracies are very high (over 90%). It is important

to notice that the data set are very small and the classifier are trained and tested per single user. This type of training it is not convenient if there will be lots of users. Because to create one model for every user will rapidly exhaust all the resources (memory, processing, storage, etc.).
The nearest neighbour approach seems able to provide also good results. It is very important to define the concept of distances between one record and another. Increasing the features, will increase the difficulty of this task. The major problem of nearest neighbour approach is the time to process new inputs that is very large as presented by Sonderen (2016).

2.1.2.6 Support Vector Machines

Support vector machines is a machine learning method. The basic idea is to find an hyperplane able to separate the classes. The method's strength is the ability to find not only a simple hyperplane but the best one, named decision boundary, among the training data. The best hyperplane is decided based on the distance between the decision boundary and the training points among all the classes which the algorithm tries to maximize. The figure 2.4 shows the best decision boundary for those sets of points.

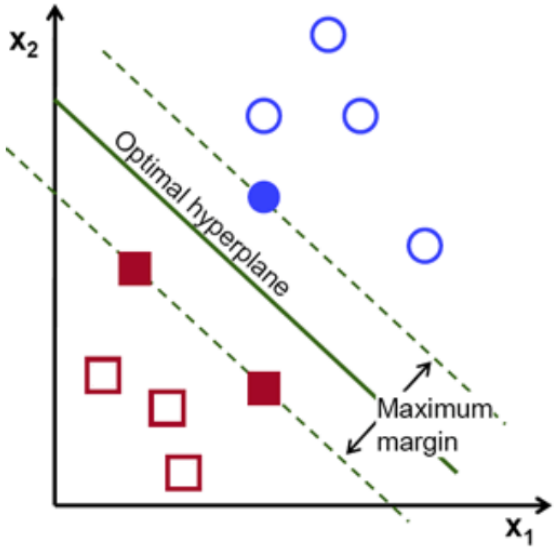

FIGURE 2.4: support vector machine best boundary

In order to classify new points, the algorithm follows a very simple rule. Using the figure as example, if the points are above the optimal hyperplane they belong to the circle class otherwise to the square class. Lots of cases have to deals with non-linearly

separable data. In these cases, the kernel trick technique is implemented. Indeed, the kernel trick transforms the training data to a new feature space that becomes linearly separable (figure 2.5).

FIGURE 2.5: support vector machines kernel trick

Cardoso et al. (2016) testes different classifiers. Along with the support vector machines, the Naive Bayes and the decision tree are also used. To do so, an accelerometer with a rate of 33 Hz is used. The Wi-Fi information are also considered to differentiate transportation modes that would have been difficult to distinguish by only using accelerometer-based features. Moreover, 21 different signals (magnitude, x y and z values, angle between the acceleration vector and each of the phone axes) are extracted from the accelerometer data. For all these signals, 17 different features (the reader may refer to the paper for the detailed list) are calculated. However, many of the features are redundant or irrelevant and have thus been discarded. The final number of features is 40. The classifiers are tested and trained using a 10-fold cross-validation [3]. The detected modes are Inactive, Walk, Metro, Train, Car and Bus.

Yu et al. (2014)'s work introduces the usage of new sensors to perform mode detection. The new sensors are the gyroscope (measure of the angular velocity) and the magnetometer (measure of the terrestrial magnetic field). The sampling rate is fixed to 30 Hz for all the sensors. The following features are extracted from the sensors mentioned above: the standard deviation of the magnitude for all the sensors, the mean of the magnitude measured by both the accelerometer and the gyroscope, the FFT peak determined by the accelerometer and the ration between the largest and the second largest FFT coefficient calculated by the accelerometer. Furthermore, the data, is divided in time windows of 512 samples, meaning 16 seconds of data is used as single input record for the classifier. As first mode detection, if the standard deviation of the accelerometer is lower than a predefined threshold, the predicted mode will be Still. The rest of the

[3]The cross-validation technique consist of dividing the training set in N different parts where N-1 parts will be used as training data and 1 as validation data (not test data). The parts are rotated to enrich the knowledge of the method.

data will be sent and classified by the following classifiers: decision tree, Adaboost and Support Vector Machines. Adaboost is an ensemble method that belongs to the boosting family (not as random forest and random subsequence methods). The principle of the boosting algorithms is to convert weak learners (classifiers with accuracy just above the random guessing) into strong learners that classify with high accuracy. Yu et al. (2014) uses the decision trees as weak learner for Adaboost. Finally, after the Adaboost classification, a voting scheme is added to correct eventual errors due to change of mode inside the time window. The predicted modes are Still, Walk, Run, Bike and Vehicle.

Fang et al. (2016) proposes new features to the dataset created by Yu et al. (2014). Since the dataset is the same in both works, the sensors and their sampling rate are the same as well. The enhancement they gave is the addition of new features calculated from the sensors. The added features are mostly calculated from the accelerometer. A list of 14 different features will be used to perform mode detection (the reader may refer to the paper for the complete list). As the previous work, the classifiers are the Support Vector Machines, the decision tree and the nearest neighbour (which replaces the Adaboost classifier). These classifiers perform two type of mode detection: the transportation mode classification which includes Still, Walk, Run, Bike and Vehicle and the vehicle mode classification which includes Car, Metro, Bus, Train and HSR (High Speed Rail). The new features added a higher accuracy in comparison to the work done by Yu et al. (2014) regarding the transportation mode detection.
Shafique and Hato (2015) proposes an approach based on the accelerometer where the device was an on purpose instrument. The accelerometer device was also able to record the minimum, maximum and average acceleration in movement, crosswise, vertical directions, resultant acceleration and average resultant acceleration. The accelerometer's sampling rate is set to 16 Hz. Although the GPS data is recorded, it is not used. After the pre-processing steps, the features were as follow: maximum, minimum and average acceleration along the three directions, differences between maximum and minimum and their differences, differences between average accelerations, resultant acceleration and average resultant acceleration. The classifiers that perform mode detection are Support Vector Machines, Decision Tree, Random Forest and Adaboost. The detected modes are Walk, Bike, Car and Train.
All the works compared lot of classifier and there are different opinions. The results are resumed in the table 2.1.

The above table (2.1) aims to let the reader understand that all the classifiers are very good. The features affect a lot the final results. In fact, it is possible to see how Support Vector Machine and Decision Tree exchanges their positions as best classifiers. It is important to notice that the works done by Yu et al. (2014) and Fang et al. (2016) are

Authors	Best classifier	Second classifier	Third classifier	Fourth classifier
Cardoso et al. (2016)	Decision tree	SVN	Naive Bayes	
Fang et al. (2016)	SVN	Nearest Neighbour	Decision Tree	
Shafique and Hato (2015)	Random Forest	Decision Tree	Adaboost	SVN
Yu et al. (2014)	SVN	Adaboost	Decision Tree	

TABLE 2.1: classifier comparison

tested on a very large dataset which makes a good accuracy really hard to achieve and thus having an overall accuracy higher than 86% is very impressive.
The Support Vector Machines are one of the most promising approaches along with the decision trees and the random forest. The definition of the features is again very important, since the approach tries to find an optimal hyperplane able to distinguish all the modes. Thus, the features should be able to separate the modes. This always remains a very difficult task.

2.1.2.7 Neural networks

Although the neural network's theory was first developed in the forties, it has only been used since the last decades thanks to the increasing processing power of computers. The basic idea is to simulate the brain's neurons and their proper connections. The network is composed of layers and each layer can contain several neurons, called Perceptron. The layers inside the network are called hidden layers. One layer's neuron output becomes the input of some other neurons in the network. All these links have their proper weights with which their respective inputs are multiplied. In the figure 2.7, a feed-forward network where the connections never form a loop nor return back to layers. In the figure 2.6, a single neuron taking 3 different inputs is represented.

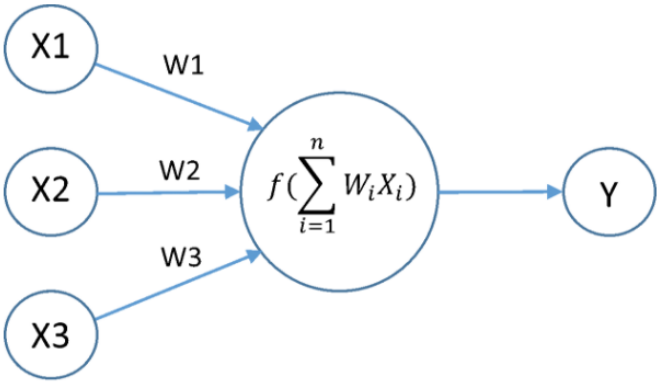

FIGURE 2.6: single neuron of a neural network

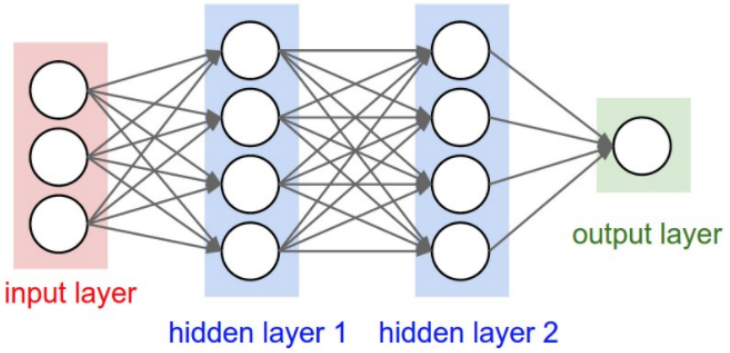

FIGURE 2.7: feed-forward neural network

The function F, called activation function, is an arbitrary function (preferably a derivable function for mathematical aspects) that calculates the output from all the inputs as shown in the figure 2.6. A possible neuron's bias can be part of the input. Regarding the learning phase, an algorithm called back propagation trains the network by changing the weights and the biases of the various connection accordingly to reduce the global error. The error, called cost function, is related to the predicted output of the network and to the real output of the training data. It is possible to define different types of errors, the most common is the mean square error.

The work proposed by González et al. (2008) uses a Multilayer Perceptron classifier on GPS data; if the GPS is not available an approximate location is calculated using the Cellular Network ID. The Multilayer Perceptron is a feedforward type neural network with at least one hidden layer. González et al. (2008) introduces an interesting approach to the data analysis. Indeed, it only saves the critical GPS points. A GPS point is considered critical if there is a change of the user's movement direction,i.e. the user's direction is not the same before and after the critical point (Figure 2.8).

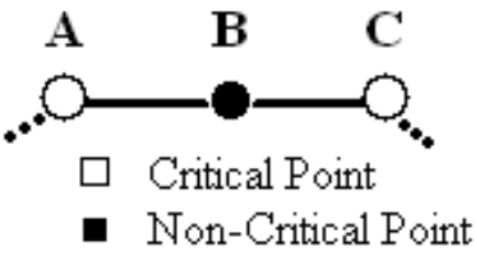

FIGURE 2.8: example of critical points

Unfortunately, the critical point approach is performed as post processing step. Thus the device records all the GPS points every four seconds. Finally, in the classification step, the authors' selected features such as the neural network input only changed if all the GPS points were used (6 different features) or if only the critical points were used (8 different features). In conclusion, the classifier was only able to detect Walk, Car and Bus.

Vu et al. (2016) proposes a new class of neural networks to cope with the mode detection problem. The classifier is called Recurrent Neural Network. The difference with the Multilayer Perceptron is the presence of cyclic loops of the neuron itself. In fact, one of the neuron's input is the output of the neuron itself. These types of connections let the hidden layers having memory allowing analysing time-sequences of data (handwriting recognition or speech recognition). Moreover, the dataset is the same as in Yu et al. (2014) work but only the accelerometer has been taken as input data. In contrast to all previous work, only the accelerometer magnitude is used. Despite the use of only one feature, the results are as good as all the previous works. The detected modes are the same of Yu et al. (2014): Still, Walk, Run, Bike and Vehicle.

Stenneth et al. (2011) use GPS data with a sampling rate of one sample every 15 seconds along with the buses real-time location data refreshed every 20 30 seconds. Indeed, the transport data led the authors to inspect new features as follow: average bus location closeness, candidate bus location closeness, average rail line trajectory closeness and bus stop closeness rate. In addition, the following GPS related features are used: average accuracy of GPS coordinates, average speed, average heading change and average acceleration. It can be referred to Stenneth et al. (2011) for more details. The used classifiers are: Naive Bayes, Bayesian Network, Decision Trees, Random Forest and Multilayer Perceptron. The classifiers are first tested with and without the transport data and finally after deletion of the unnecessary features, i.e. the ones considered as less predictive. The elimination process is performed by two different algorithms: Chi Squared and Information gain. The detected modes are: Still, Walk, Bike, Car, Bus and Train.

Stenneth et al. (2011) compare various classifiers which all proved to have an overall accuracy of 90%, except for the Multilayer perceptron which has an accuracy of 83%. The classifier with the highest accuracy is the Random Forest approach (few points compared to the others). González et al. (2008) show good results for the Multilayer Perceptron with the critical points reaching an accuracy of 90%. Using all the GPS points gives

lower results, proving that the critical points technique is very interesting. It is unfortunate however that, for the moment, it is only a post processing step. Finally, Vu et al. (2016) work is very impressive; just using one feature they got over 90% accuracy over a very large dataset, obtaining the same results as Fang et al. (2016) and Yu et al. (2014).

The neural network technique is also very promising. The major advantage is that it is able to find some information that is not visible to humans (González et al., 2008). Vu et al. (2016) show that only one feature is enough to provide impressive results; although, it is important for this approach to define the good features.

2.2 Sensors

Choosing the right sensor is extremely important in mode detection. Thus, several variables should be considered before selecting the sensors. The table 2.2 summarises all the investigated sensors and the classifiers used to perform mode detection.

Looking at table 2.2,it is possible to notice that the accelerometer and the GPS are the most used sensors. The accelerometer sensor calculates the device's acceleration along the three axes. The GPS sensor is able to retrieve the user's location in terms of longitude and latitude.

Analysing the literature, it is possible to observe that the accelerometer has no common sampling rate; each research work defined its own sampling rate. On the contrary, the GPS has a tendency for one sample every second. The accelerometer's rate needs to be sufficiently high to recognize the mode. For example, the steps while walking or the legs movement while biking. Regarding the GPS, the rate should be suitable to obtain reliable data to calculate the features (in most cases the speed). Low sampling rate, in both sensors, will results in high misclassification due to lack of data.

Occasionally, the above sensors are supported by others sensors, Wi-Fi, Cellular Network, Bluetooth and GTFS applications, that are never used alone to perform mode detection. Those supporting sensors are used to improve the user position when the GPS is not properly working (no sufficient satellites in sight). GTFS information play an important role in classification activities (Bucher et al. (2016) and Stenneth et al. (2011)) but they are related to specific cities; thus, GTFS, to correctly work everywhere, it is mandatory to have information from each city. This constraint is the major disadvantage of GTFS information. A similar problem is encountered by the Bluetooth

Authors	Classifier	Accelerometer	GPS	Other sensors
Nitsche et al. (2012)	Random Subspace Method	100 Hz	1 Hz	
Geurs et al. (2015)	Bayesian	X	2 Hz	Wi-Fi and Cell id info
Bucher et al. (2016)	Bayesian	Moves app	Moves app.	Open street map
Montoya et al. (2015)	Bayesian	20 Hz	1 Hz	Wi-Fi, Bluetooth and Cell ID info
Manzoni et al. (2011)	Decision Tree	25 Hz	1 Hz	Occasionally WIFI
Montini et al. (2015)	Decision Tree	5 Hz	1 Hz	For location also WIFI
Xiao et al. (2012)	Decision Tree	N.A	N.A	GSM
Wang et al. (2017)	Decision Tree/Rnd. forest	X	1 Hz	X
Sauerlnder-Biebl et al. (2017)	Fuzzy	X	0.25 Hz	X
Schüssler and Axhausen (2009)	Fuzzy	X		X
Shin et al. (2015)	Fuzzy	1 Hz	X	Network based location
Widhalm et al. (2012)	Decision tree followed by HMM	50 Hz	1 Hz	Cell. Net for location
Sonderen (2016)	Decision tree, random forest and Nearest Neighbour	140 Hz.	X	X
Reddy et al. (2008)	Decision tree followed by HMM	35 Hz	1 Hz	X
Martin et al. (2017)	Movelet approach, k-NN and random forest	5 Hz	1 Hz	X
Cardoso et al. (2016)	Naive Bayes, SVM and decision tree	33 Hz	X	X
Stenneth et al. (2011)	Bayesian net, Decision Tree, Random forest, Naive Bayes and MLP	X	1 every 15 Seconds	GIS info regarding bus stops
Fang et al. (2016)	Decision tree, k-NN and SVM	30 Hz	X	Gyroscope and magnetometer (30 Hz)
Shafique and Hato (2015)	SVM, Adaboost, Random forest and decision tree	16 Hz	collected but not used	X
Yu et al. (2014)	Adaboost, decision tree and SVM	30 Hz	X	Gyroscope and magnetometer (30 Hz)
González et al. (2008)	Neural network	X	0.25 Hz	Cell ID (when there is no GPS)
Vu et al. (2016)	RNN	30 Hz	X	X

TABLE 2.2: list of sensors and classifiers used by the authors

sensor; to properly work, the Bluetooth needs to communicate with beacons that should be spread all over the city. Others sensors like the gyroscope and the magnetometer have been tested by Yu et al. (2014), Fang et al. (2016) and Vu et al. (2016) and allow improving the accelerometer classification. The disadvantage is that such sensors are not always available in smartphones, while the accelerometer is almost always available. Another important aspect is the device's battery life. Montini et al. (2015) compare the

battery drain issue of a smartphone using the GPS or the accelerometer. The GPS sensors needs lots of energy to calculate the user's position, while the accelerometer sensors need much less energy to collect the data. In conclusion, if the battery life is important, the GPS is not recommended, while the accelerometer is battery friendly.

Chapter 3

Objectives and Methodology

The thesis has been carried out within the research work of the Chair MIDT (Mobilité Intelligente et Dynamiques Territoriales) in which a smartphone app, *Mobilité Dynamique*, has been developed. The app is an Android application (the iOS version will be soon released) that allows the users to keep track of their mobility patterns. Several statistics let the users understand their travel behaviours such as the number of daily trips and the means used for travelling. Moreover, a heat map shows the most visited or favourite places. The application is user friendly as it only requires the GPS sensor to work in the background (the smartphone's screen is off or another application is running in foreground). Furthermore, a section where the users can chat and share news about the traffic, car accidents or buses delay is also available. Users are allowed to register the trips by themselves, by selecting the transport mode they are going to use. The trip is stored in memory, by simply clicking on the check in button to start the trip and on the check out button once arrived at destination. This action confirms to the application the used mode. The figures 3.1, 3.2 and 3.3 show some of the application's screens.

FIGURE 3.1: application's home page

FIGURE 3.2: statistics of movements

FIGURE 3.3: application's trips map

The application is able to recognize the following modes: Still (no movement), Walk, Bike and Vehicle (car, bus, train etc.). In order to perform the mode detection, the Google APIs (Google APIs 2017) are used.

Only low power sensors (not specified by Google) are used to retrieve the mode in order to save battery. Cardoso et al. (2016) performed a test on the APIs and showed that the on vehicle method does not have a good accuracy. The same problem is related to the *Mobilité Dynamique* application. In the figure 3.4, a train trip (from Paris to Bruxelles) has not been recognized and, thus, classified as unknown. This is a problem that needs to be solved.

FIGURE 3.4: The application's fail on mode detection

The *objective* of the thesis is to design a smartphone API able to perform transport mode detection. The API which will be integrated in Mobilité Dynamique, will have to rely on smartphone sensors to recognize in real-time the user's mode. The most important requirements are: high accuracy in mode detection and low energy consumption. Both requests are essential if we want the application is actually used. Moreover, since we want all users to take advantage of the application features, the API should run on all smartphones models or as much as possible.

3.1 Methodology

In order to perform mode detection, a classification algorithm has been designed. As already mentioned in section 2.1, a classification algorithm F tries to assign to a predefined set of classes or categories Y a new input data X:

$$y = f(x)$$

The classification algorithm F needs to be instructed in order to be able to assign the new input to the correct class. To do so, a training dataset is necessary. To verify the correctness of the trained classifier, a test dataset is needed as well.

To this end, a methodology has been defined, articulated in five steps: a) choice of the sensor used to perform mode detection; b) the selection of the data source (training and test data) to train the classifier; c) the definition of classifier; d) the learning phase and e) the definition of the algorithm to detect the modes.

3.1.1 Sensor choice

The state of the art section presented various sensors that have been used to perform mode detection. As mentioned, the most used sensors are the accelerometer and the GPS. Both sensors, whether used together or separately, reach very good scores in overall accuracy. Since the condition on the accuracy can be satisfied by both sensors, the final decision is based on the energy consumption. In section 1.2 the battery problem is analysed and thanks to the considerations made by Montini et al. (2015) the final choice is the accelerometer. The accelerometer sampling rate will be discussed in the next subsection.
The accelerometer will not be supported by other sensors for the following reasons (refer also to section 2.1):

- the Wi-Fi, the Bluetooth and the Cellular network are almost always used as support to the GPS in order to improve the estimation of the user's location. Since the GPS will not be used to perform mode detection, the other sensors will not be used as well;
- the Gyroscope and the Magnetometer are not as common in smartphones as the accelerometer, thus they will not be used;
- the GTFS and GIS information are related to the specific city. The large amount of data needed to take advantage of this information and to be operational everywhere, makes the use of GTFS and GIS unpractical.

3.1.2 Selection of the data source

The data source selected for the training is the dataset created by Yu et al. (2014). [1]

The dataset is a complete set of training and test data. The data was recorded with the HTC One mobile phone and was provided by users with different demographics characteristics (gender and age). The collected transport modes are: Still, Walk, Run, Bike, Motorcycle, Car, Bus, Metro, Train, and high speed rail (HSR). For each mode,

[1] Sir. MengChieh Yu kindly provided me with the rights to download the dataset for research purpose only.

the smartphone that collected the data was located in different positions pockets, bag or hands etc. The data were recorded by the accelerometer, the gyroscope and the magnetometer sensors, each with a sampling rate of 30 Hz. More than 20 Gigabyte of log files were used to train the classifier.
The source data permits to perform supervisioned learning as the data had already been assigned labels i.e. it is known the mean of transportation which the data belongs to.

3.1.3 Recurrent Neural Network

Several classifiers are presented in the state of the art. Most of them proved to have good results such as Support Vector Machines, Decision Tree or Neural Networks. Previous projects proposed different features with which the classifiers are trained.
Since the classifier will be used on a smartphone, it is convenient not to overwhelm the mobile with several calculations in order to retrieve the features as this will quickly drain the battery. The work done by Vu et al. (2016) solves this issue. In fact, only the accelerometer is used and only one feature is calculated. The suggested classifier was tested on the data source described above and achieved satisfying results.
The classifier selected for this work is a Recurrent Neural Network (RNN). As already explained in the state of the art (2.1.2.7), the RNN is an application of the Neural Network which peculiarity is the presence of back loops; for more details see Olah (2015). The figure 3.5 shows the basic structure of a RNN: the block A is the Neural Network where the output h_t will become the new input along with a new input record x_t.

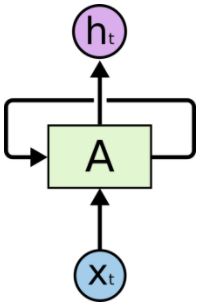

FIGURE 3.5: Basic structure of a RNN

The back loops are the strength of the RNN. In fact, the algorithm will be able to interpret the new input based on the past records already treated by the network. Let's take as an example the following phrase:

I come from Italy so I can speak Italian.

Based on the first words the network will be able to suggest the correct final word. In this example, each word feed the input of the RNN step by step. Figure 3.6 shows the unrolled version of a RNN adapted to the example:

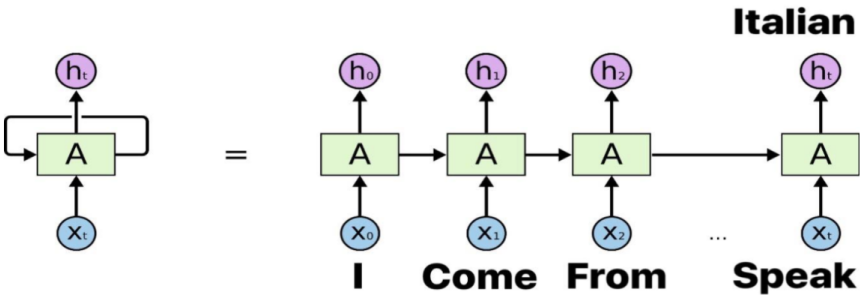

FIGURE 3.6: unrolled RNN with the example

Looking at Figure 3.6, it is possible to understand that RNN is designed for time-series or sequences of data. The different numbers of block A represent the maximum number of elements in a sequence among all the sequences used in the training phase. It represents one of the most important parameters of a RNN (it will be discussed later). An incurring problem is the length of the sequences. In fact, if the sequence is too long the RNN is no longer able to correlate distant inputs. In such situation, one possible solution are the LSTM (Long Short Term Memory) networks. They replace the A block with a *Cell state* C_t (figure 3.5). Thus each LSTM network is able to add or remove information that will affect the output and the following LSTMs network (figure 3.10). The input provided to the *Cell state* is controlled by structures called *Gates*. There are three different *Gates*: the forget gate (figure 3.7), the input gate (figure 3.8) and the output gate (figure 3.9). The *Gates* take as input the new input record X_t and the output of the previous layer h_{t-1}. The forget gate defines how much information should be kept from the *Cell state* C_{t-1} coming from a previous layer. The input gate defines how much information should be added to the *Cell state* already processed by the forget state. The output gate defines the output h_t of the LSTM cell which also uses the results of the *Cell state* C_t which was processed by the two previous *Gates*.

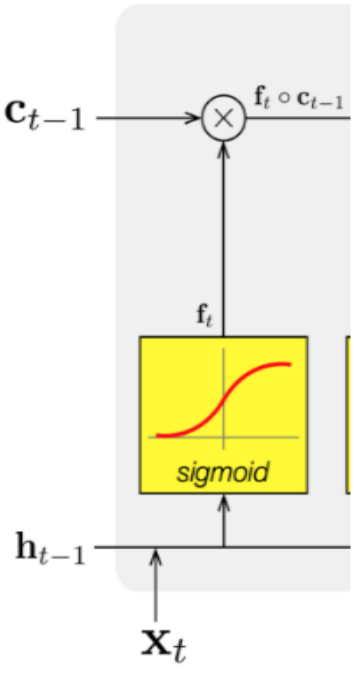

FIGURE 3.7: the forget gate

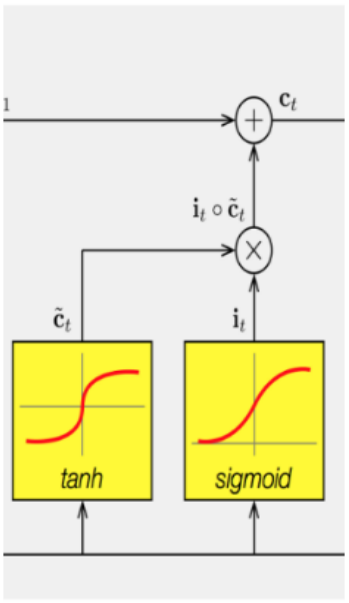

FIGURE 3.8: the input gate

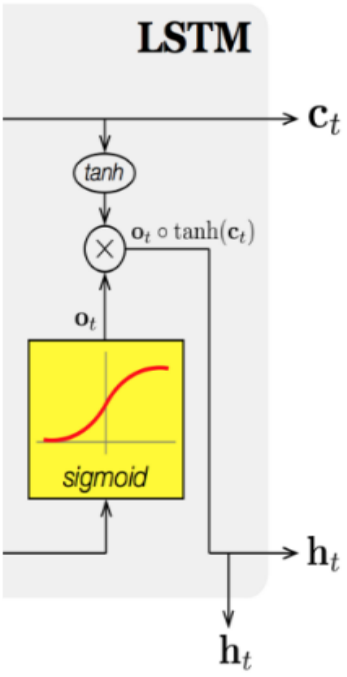

FIGURE 3.9: the output gate

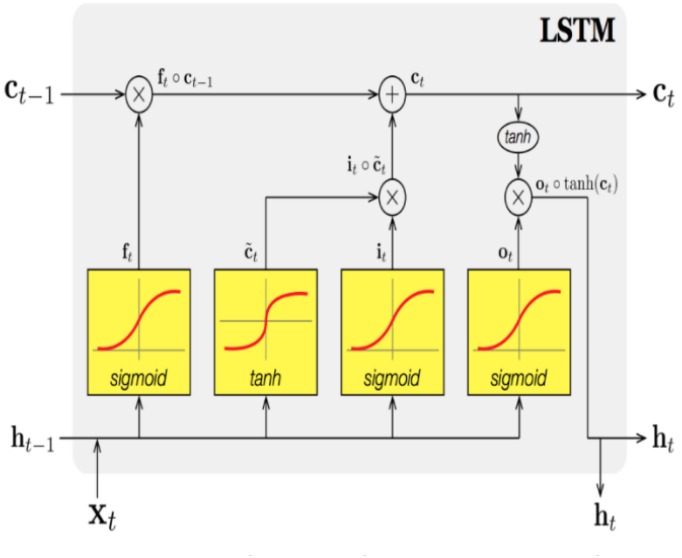

FIGURE 3.10: the complete LSTM network

In the figure 3.10, the yellow squares are pre-trained neural network layers that implement the sigmoid and the hyperbolic tangent. One of the parameters of the RNN is the dimensionality of these neural networks, i.e. the number of neurons. The output's dimensionality of the LSTM network will have the same order of magnitude. Higher dimensionality means higher RNN complexity.

To resume, the LSTM network is a predefined module with which it is possible to define the dimensionality. The RNN is a concatenation of LSTMs network where the cardinality depends on the number of elements of an input sequence. Each element of the sequence is the input of the appropriate LSTM network (figure 3.6).

3.1.4 The learning phase

Regarding the training phase, the algorithm looks to the output of interest, that is the LSTM network output with an input coinciding with the last input of the sequence. The output of interest will become the input of a neural network that needs to be trained, for example learning the appropriate weights and biases. In figure 3.11 it is possible to see that the Softmax block represents the neural network that will be trained to calculate the correct output.

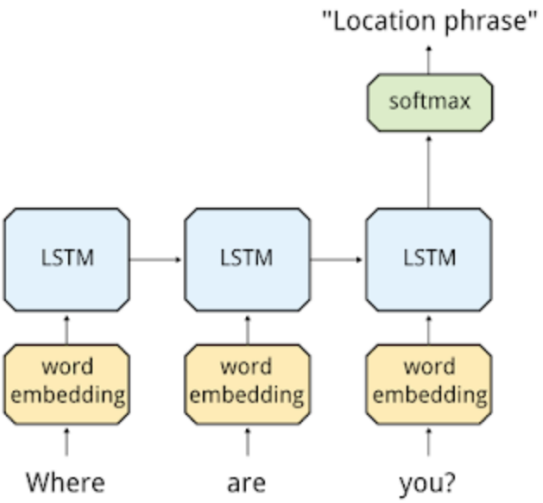

FIGURE 3.11: the softmax block that needs to be trained

Softmax is the Cost function that calculate the loss that needs to be minimized. The algorithm will train the weights and the biases of the neural network in order to minimize the Cost function. The Cost function takes the predicted output and compares it to the correct output for all the sequences. The loss decreases if the RNN is able to predict the correct output. This computation is performed multiple times in order to reach a good prediction accuracy, that is a suitable loss threshold.

3.1.4.1 Python and TensorFlow

The programming language used to implement the RNN is Python, version 3.6, through the open-source machine learning framework TensorFlow (version 1.4). TensorFlow is a suite developed by Google which provides all the tools to implement most of the machine learning algorithms. TensorFlow runs on CPU (Central Processing Unit), but it is also able to exploit the GPU (Graphics Processing Unit) which makes the training phase faster since because of the higher computational capacity.

The workstation details are:

- HP Z440 Workstation;
- Processor Intel Xeon 3.5 Ghz 4 cores;
- Ram 32 Gb;
- GPU Nvidia Quadro M4000.

3.1.4.2 Experimental Settings

The source data, as discussed in the subsection 3.1.2, provides various log files for all the transport modes. The first important step is to define the length of the sequences that will be the input to the RNN. According to the work done by Vu et al. (2016), each sequence is composed of 350 accelerometer magnitude values. Since the sampling rate is set to 30 Hz, the series of elements in the sequence corresponds approximately to 12 seconds. This timing is a tradeoff between the necessity of performing real-time prediction and accurately recognizing the mode of transport.

As explained in the previous section, the number of concatenated LSTM networks depends on the number of elements in a sequence. The input of a LSTM networks is the arbitrary element in a sequence which can be grouped together to create a single input element for the LSTM networks. Figure 3.12 shows an example of how a sequence can be divided.

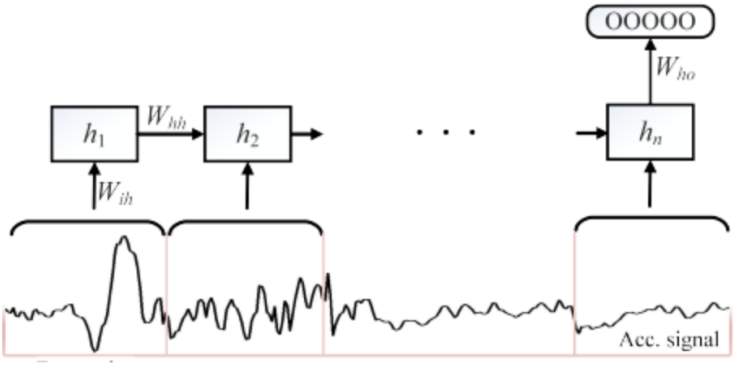

FIGURE 3.12: representation of the input of a RNN (Toan et al. (2016))

The following LSTM network inputs are tested:

- 1 accelerometer sample per group, meaning 350 LSTM networks concatenated;
- 5 accelerometer samples per group, meaning 70 LSTM networks concatenated;
- 10 accelerometer samples per group, meaning 35 LSTM networks concatenated.

Another parameter to be tested is the LSTM network dimensionality or hidden layers. Higher dimensionality means higher computation and complexity, thus it is not always the best choice. The tested values are the following:

- 64;
- 128;
- 256.

One parameter that was not mentioned in 3.1.3 is the batch size. The batch size is not related to the RNN theory, but is related to the computation of the Cost function in terms of programming aspects. It represents the number of sequences used to perform a training phase. When all the sequences are used, an epoch has been completed. After that, the sequences are reused to train again the classifier. This parameter must be considered because otherwise the resources in terms of memory will be immediately exhausted due to the size of the dataset. The selected batch size for this work is set to 380 sequences for each step of the training phase.

3.1.5 Mode detection

The modes of transport that the algorithm is able to classify are the following:

- Still;
- Walk;
- Bike;
- Motorcycle;
- Car;
- Bus;
- Train.

Once the training phase is complete, the algorithm, is able to recognize the mode of transport given a sequence of the accelerometer sample data. To test the algorithm on real data, an Android application has been developed.

The TensorFlow framework trains a RNN model to perform the mode detection. After the training phase, the model is ready to be deployed. The TensorFlow suite allows to save the model and use it on an Android application. Through the Android Studio software environment, an application has been developed to perform mode detection. The application collects enough accelerometer data to construct a sequence then it performs mode detection through the model trained by the RNN. Since the model is already trained, the classification is instantaneous. The data flows in the LSTMs and the last LSTM output pass through the trained neural network (Softmax block) which prints the recognized mode.

The modes that are tested using a smartphone (Samsung Galaxy S3 GT-I9300 with android 4.3) are the following:

- Still;
- Walk;
- Bike;
- Car;

- Bus;
- Train.

The mode detected is printed on the smartphone screen.

Chapter 4

Results

This section aims at evaluating the training performance of the classifier, thus its effectiveness to accurately perform mode classification with real data. The RNN's training phase results are presented in the Classifier evaluation section. Both a confusion matrix [1] and an accuracy result are used as performance metrics of the classifier. The Application evaluation shows the smartphone's classification for each mode of transport (except for the motorcycle). Finally, the section Discussion comments and explains the results.

4.1 Classifier evaluation

The classifier is tested with the experimental settings presented in section 3.1.4.2. The different input size parameters are 1, 5 and 10. Each input size paramenter is tested with different dimensionality corresponding to 64, 128, and 256.

4.1.1 Input of size 1

As provided in the list of experimental settings, the first tested input has a size equal to 1. Unfortunately, if the LSTM's dimensionality setting is equal to 64, 128 or 256,the GPU runs out of memory due to the large allocation requirements, thus it was not possible to test the classifier's performance.

[1] The confusion matrix is a tool used to visualize the performance of a classifier. The rows correspond to the predicted class and the columns to the correct ones.

4.1.2 Input of size 5

The second input size tested is that proposed by Vu et al. (2016). Since the RNN, with LSTM dimensionality set to **64**, was not able to converge, it was not possible to find a suitable model to perform the mode detection on the training data. Figure 4.1 shows how the classifier was not able to find a suitable minimum. The classifier tried to restart from scratch several times but the loss never reached the loss threshold which was set to stop the training phase.

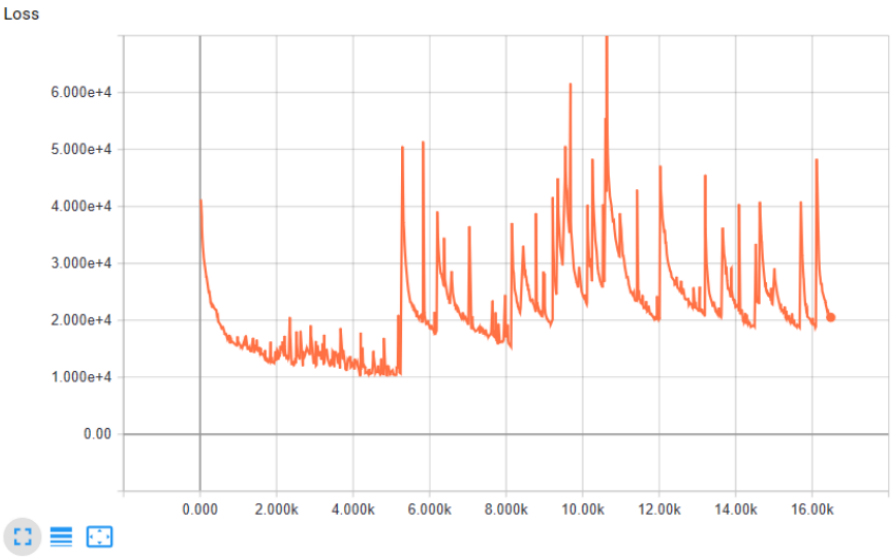

FIGURE 4.1: Loss value over time for input size 5 and dimensionality 64

The second LSTM dimensionality is set to **128**. The overall accuracy is equal to 0.658 and the confusion matrix is represented in Table 4.1.

Predicted class \ Correct class	*Still*	*Walk*	*Bike*	*MotCyc*	*Car*	*Bus*	*Train*
Still	**360**	0	0	7	176	39	127
Walk	5	**447**	0	7	18	9	1
Bike	1	33	**409**	81	10	32	10
MotCyc	0	2	68	**328**	14	35	3
Car	62	0	7	89	**453**	87	125
Bus	0	0	1	10	5	**2**	0
Train	67	1	11	19	84	128	**648**

TABLE 4.1: Input of size 5 and dimensionality 128

The *confusion matrix* can be read as follow: the units inside the table are the sequences generated by the test dataset. Each sequence had passed through the classifier which predicted a class (rows). Based on the sequence correct class (columns), the sequence is placed inside the matrix. For instance, if the classifier predicts the class Bike, but the correct class of that specific sequence is Bus, the cell [Bike, Bus] is increased by one unit. The objective is to have the highest possibles values in the cells corresponding to the same class.

The third dimensionality is set to **256**. The overall accuracy is equal to **0.677** and the confusion matrix is represented in Table 4.2.

Predicted class \ Correct class	*Still*	*Walk*	*Bike*	*MotCyc*	*Car*	*Bus*	*Train*
Still	**303**	0	3	9	201	56	137
Walk	1	**461**	0	6	2	17	0
Bike	1	12	**474**	48	14	18	9
MotCyc	3	8	56	**334**	19	28	2
Car	68	1	6	101	**497**	69	81
Bus	4	0	2	19	18	**1**	0
Train	61	0	13	26	90	97	**671**

TABLE 4.2: Input of size 5 and dimensionality 256

4.1.3 Input of size 10

The RNN, with LSTM dimensionality set to **64**, was not able to converge, it was not possible to find a suitable model to perform mode detection on the training data (as happened to the input size set to 5). Figure 4.2 shows how the classifier was not able to find a suitable minimum and stopped after the limit of 30000 epochs was reached.

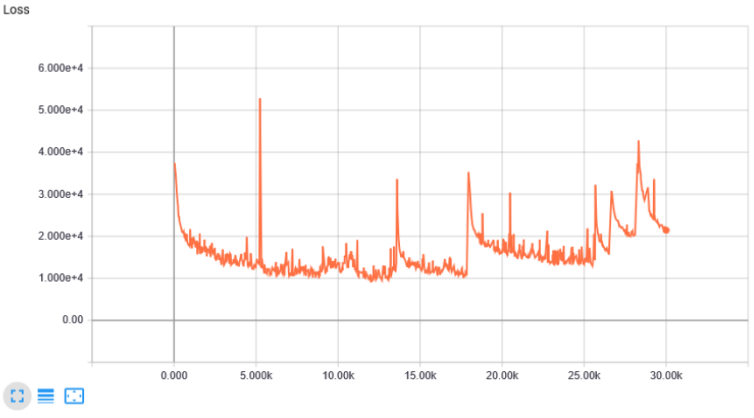

FIGURE 4.2: loss value over time for input size 10 and dimensionality 64

The second dimensionality is set to **128**. The overall accuracy is equal to 0.625 and the confusion matrix is represented in the Table 4.3.

		Correct class						
		Still	*Walk*	*Bike*	*MotCyc*	*Car*	*Bus*	*Train*
Predicted class	*Still*	**369**	0	1	7	117	80	135
	Walk	1	**437**	0	14	8	25	2
	Bike	3	46	**391**	91	12	24	9
	MotCyc	0	12	87	**285**	35	30	1
	Car	81	1	18	69	**430**	91	133
	Bus	2	1	3	17	15	**3**	3
	Train	65	1	3	22	100	152	**615**

TABLE 4.3: Input of size 10 and dimensionality 128

The third dimensionality is set to **256**. The overall accuracy is equal to 0.655 and the confusion matrix is represented in Table 4.4.

		Correct class						
		Still	*Walk*	*Bike*	*MotCyc*	*Car*	*Bus*	*Train*
Predicted class	*Still*	**398**	0	0	11	152	58	90
	Walk	1	**411**	0	11	32	31	1
	Bike	4	32	**450**	57	9	18	6
	MotCyc	1	5	79	**313**	24	28	0
	Car	57	1	11	89	**433**	107	125
	Bus	3	0	3	13	22	**2**	1
	Train	67	1	5	27	100	112	**646**

TABLE 4.4: Input of size 10 and dimensionality 256

4.2 Application evaluation

To evaluate the application, the chosen model was the one with input data of size equal to 5 and dimensionality equal to 256 (see section 4.3 for further details). The results are presented through the smartphone's screenshots. The screenshots show a list of the detected modes. Each line represents the detected mode once enough accelerometer data (12 seconds) had been collected. The value beside the detected mode represents the model confidence in its prediction. If the model does not have enough confidence (less than 1) for each class, a question mark is printed stating that the model is not able to recognize a mode.

4.2.1 Still

From the figure 4.3, it is possible to observe that the still mode is classified either as Still or as Train. The car mode is rarely suggested.

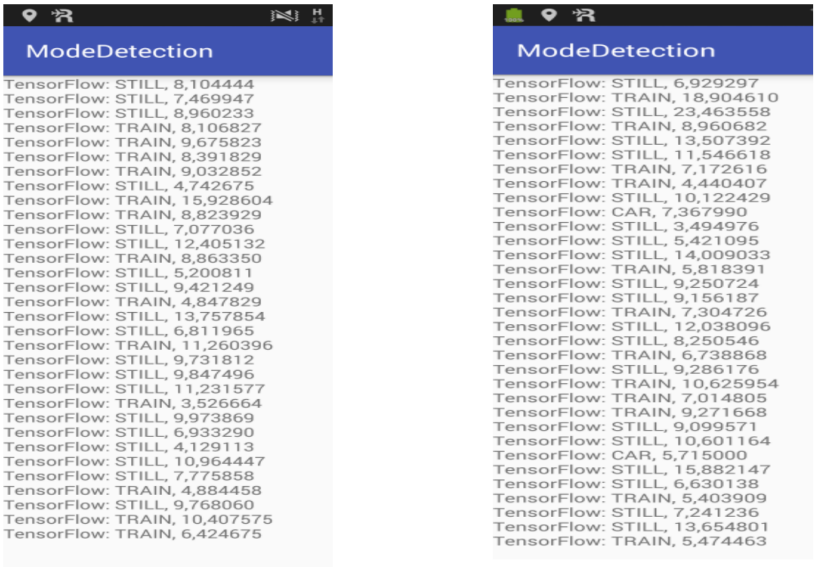

FIGURE 4.3: results of the still mode

4.2.2 Walk

From Figure 4.4 shows that the classifier has no doubt about the recognized mode except for few misclassifications. It is important to mention that the smartphone was held in the pocket of the jacket while walking.

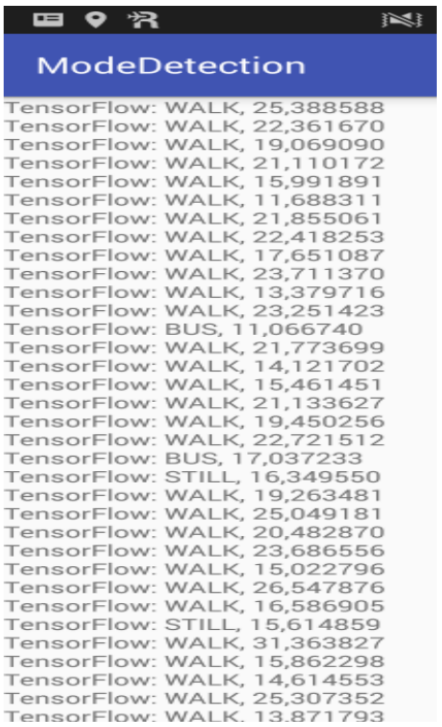

FIGURE 4.4: results of the walk mode

4.2.3 Bike

In Figure 4.5, the classifier tries to recognize a bike session. Both the start and the end of the list are misclassified because they represent the phase of either getting on or off the bike and there was no real movement yet. During the bike session the classifier has no doubt about the mode. The smartphone was carried inside the trouser pocket.

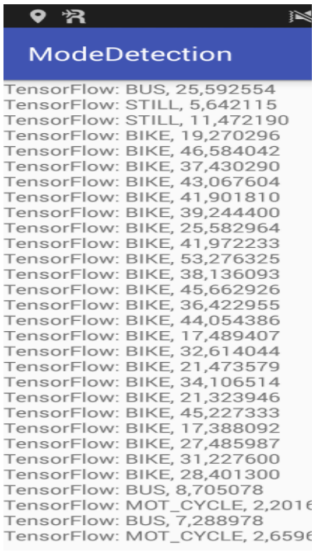

FIGURE 4.5: Results of the bike mode

4.2.4 Car

Figure 4.6 shows that the classifier is trying to recognize a Car session. It is possible to notice that there is a confusion with the bus because the acceleration profiles of the car and the bus are very similar. The car is also misclassified with the train mode. This misclassification occurs when the car does not move, for instance when the car stops because of traffic lights, traffic congestion or the car's speed is constant.

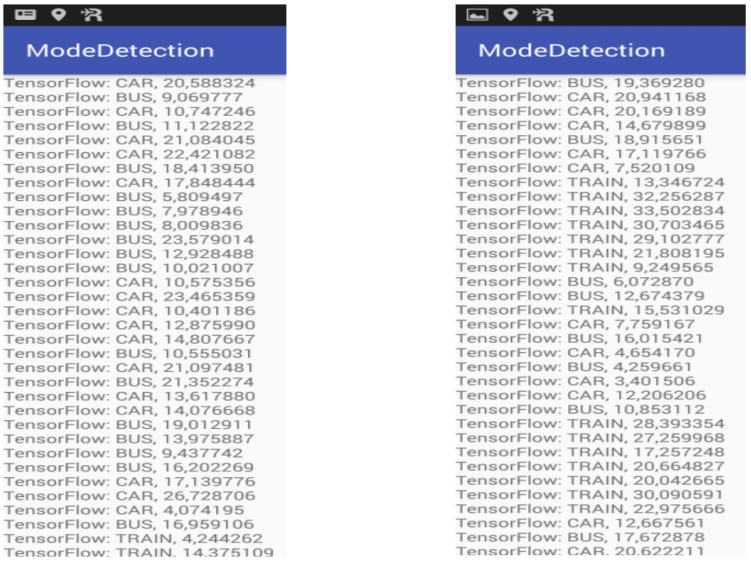

FIGURE 4.6: Results of the car mode

4.2.5 Bus

Figure 4.7 represents the classifier trying to recognize a Bus session. Unfortunately, since the task is very complicated, there are several misclassifications. The presence of the Still and Train classes is due to the fact that the bus has to stop to let passenger get on and off. The Car and Motorcycle are also suggested instead of the bus from time to time because of their similar acceleration profiles. In Figure 4.7, it is possible to conclude that the bus is the actual detected mode because it is suggested more often than the other modes. The bus stops complicate the bus classification procedure.

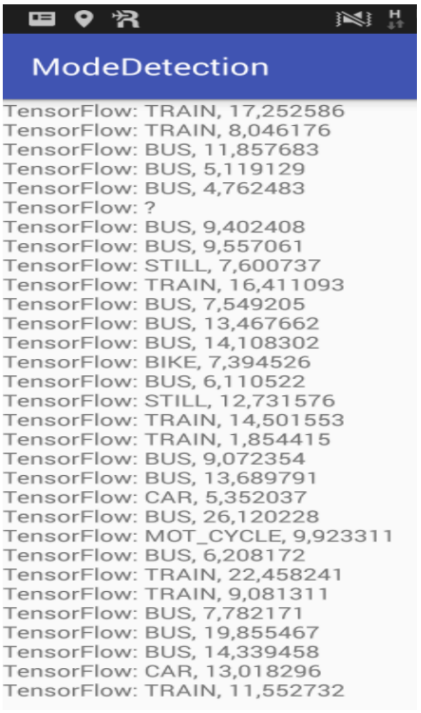

FIGURE 4.7: Results of the bus mode

4.2.6 Train

Figure 4.8 shows the classifier results of a Train's session. When the train runs smoothly (no excessive vibrations), the classifier perfectly detects the class. However, when the vibrations become stronger, the classifier has troubles detecting the right class. On the right side of Figure 3.8, the first set of misclassifications happened when the train stopped at the train station. The second set of misclassifications is due to the fact that the passenger was walking while the train was moving.

FIGURE 4.8: Results of the train mode

4.3 Discussion

In this sections are reported the comments on the results of the classifier and the Android application just presented.

4.3.1 Classifier

Analysing Tables 4.1, 4.2, 4.3 and 4.4 of section 4.1, it is possible to observe that there are numerous misclassifications and that the trend of the various experimental settings is very similar.

The Still mode is confused several times with the Car and Train. The reason is that, when the motorized vehicle reaches a constant velocity, there is no acceleration, thus the classifier has some difficulty to distinguish the three modes. To better distinguish the Still mode from the motorized modes, the accelerometer should be very sensitive to detect the engine vibrations of the car/train. The input of size 10 has better results in still's classification, especially if the LSTM has a dimensionality equal to 256 (table 4.4).

The Walk mode had only few misclassifications. An input of size equal to 5 is much more precise than an input of size equal to 10 and has the highest accuracy for a dimensionality equal to 256 (table 4.2).

The classification of the Bike mode was quite good even if the motorcycle mode was suggested for a couple of times. This is due to the similitude of the two transport modes. An input of size equal to 5 and a 256 dimensionality proved to be the best experimental setting (table 4.2).

The Motorcycle class has misclassifications with the Bike mode due to the similarity of the two modes of transport. Few misclassification occur with the other motorized modes but they are negligible. This is due to the fact that the acceleration profile of these mode is sometimes similar.
The Car class was misclassified several times and all other motorized modes were wrongfully suggested, even the still class. The same trend of misclassification is similar for the Bus and Train modes. The reason is that, unlike the walk and bike modes, the motorized modes do not have an acceleration pattern and, thus, the Bus and Car tend to have a similar behaviour. The misclassification with the train probably happens when the car and the bus reach constant velocity, that is acceleration equal to zero. This situation makes the acceleration sequence of the two modes of transport very similar. The best experimental settings for all the classes is an input of size equal to 5 and a LSTM dimensionality equal to 256 (table 4.2).
Unfortunately, the test data for the Bus class were not sufficient. The reason is that the log files do not contain enough data to create sequences of 350 samples. However, the training data is sufficient and the bus was correctly detected, as it was mentioned in the subsection 4.2.5.
In comparison to the previous works, presented in the state of the art analysis (section 2), the overall accuracy is lower. The main reason for this result is attributed to the fact that, within this thesis work, the misclassification of the still class is taken into account. However, the misclassification of the motorized modes is still a current issue and no reliable solutions have been provided yet.

The classifier was also tested by creating two sets of classes:

- Still, Walk, Bike and On Vehicle;
- Motorcycle, Car, Bus and Train.

Since the configuration did not generated better results compared to the discussed experimental settings, no further investigation has been made. Another test was also performed by adding the Gyroscope samples in addition to the acceleration samples. Even if more information was provided to the classifier, the accuracy was lower in comparison to the accuracy obtained by only using the accelerometer data. Therefore, no further investigation has been made.

4.3.2 Application

The model used to test the application is characterised by an input of size equal to 5 and a LSTM dimensionality equal to 256. The reason behind this choice is that this model

proved to have the highest overall accuracy and it correctly classified the highest number of sequences (except for the still class) for all the classes. The reason to prefer a higher classification accuracy for the motorized modes despite the Still mode is explained in the conclusion chapter.

Although the classifier does not generate excellent results, the application is still able to distinguish some classes. For certain modes, however, it does not have a good accuracy (notably Car and Bus).

It is not enough to rely on one sequence to accurately detect a class. Instead, it is necessary to rely on several sequences. For instance, if a car stops because of a red light and only one sequence is used, the classifier will probably wrongfully suggest the train mode. At least 8 to 10 sequences are needed in order to accurately assign a mode. Thus one minute and a half to two minutes of acceleration measurements are required.

Chapter 5

Conclusion

The thesis work aims at finding a method to perform mode detection with high accuracy while saving the battery life. The use of the accelerometer sensor alone is the best choice to perform mode detection, while keeping in mind the battery life. Unfortunately, the accuracy is affected by this choice and it is not sufficient to rely on this type of configuration to properly define the transport mode. This work ought to be the first step to construct a better classifier by improving the classification. The main challenge is to reduce the number of misclassification. To do so, it is important to understand the major causes which lower the overall accuracy:

- The misclassifications between the Still class and the motorized modes;
- The misclassifications between the motorized modes (mostly car and bus).

Since the *Mobilité Dynamique* already uses the GPS sensor to track the user's travel patterns, it could be very interesting in the future to rely on both the GPS sensor and the accelerometer to increase the overall accuracy of the classifier. For instance, in order to distinguish the Still and the motorized modes (notably the train), it is simply necessary to check if the user has moved during two GPS records. If there is no movement, the assigned class should be Still. On the other hand, if a movement is detected then the assigned class should be a motorized mode.
Regarding the motorized modes, the classifier does not confuse these classes with the Walk and Bike modes which are detected with high confidence. To increase the accuracy of detecting the motorized modes, an idea could be to increment the accelerometer sampling rate in order to have more details about the different acceleration profiles of the modes. It is important to keep in mind that this decision will affect the battery life.
To conclude, I am really satisfied with this work because I believe that we are closer than ever to successfully perform mode classification with reliable accuracy. Moreover,

this research gave me the possibility to learn more about the transport world and to understand its complexity. Besides, I had the possibility to improve and fine-tune my knowledge of Artificial Intelligence, TensorFlow and Android applications.

Bibliography

Bucher, D., Cellina, F., Mangili, F., Raubal, M., Rudel, R., Rizzoli, A. E., and Elabed, O. (2016). Exploiting fitness apps for sustainable mobility - challenges deploying the goeco! app. In *Proceedings of the 4th International Conference on ICT for Sustainability (ICT4S)*, Advances in Computer Science Research, pages 89 – 98. Atlantis Press. 4th International Conference on ICT for Sustainability (ICT4S); Conference Location: Amsterdam, Netherlands; Conference Date: August 29 - September 1, 2016; .

Cardoso, N., Madureira, J., and Pereira, N. (2016). Smartphone-based transport mode detection for elderly care. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6.

Fang, S.-H., Liao, H.-H., Fei, Y.-X., Chen, K.-H., Huang, J.-W., Lu, Y.-D., and Tsao, Y. (2016). Transportation modes classification using sensors on smartphones. *Sensors*, 16(8).

Geurs, K. T., Thomas, T., Bijlsma, M., and Douhou, S. (2015). Automatic trip and mode detection with move smarter: First results from the dutch mobile mobility panel. *Transportation Research Procedia*, 11:247 – 262. Transport Survey Methods: Embracing Behavioural and Technological Changes Selected contributions from the 10th International Conference on Transport Survey Methods 16-21 November 2014, Leura, Australia.

González, P. A., Weinstein, J. S., Barbeau, S. J., Labrador, M. A., Winters, P. L., Georggi, N. L., and Perez, R. (2008). Automating mode detection using neural networks and assisted gps data collected using gps-enabled mobile phones. In *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting*, pages 1–12.

Manzoni, V., Maniloff, D., Kloeckl, K., and Ratti, C. (2011). Transportation mode identification and real-time co2 emission estimation using smartphones how co2go works. Technical report, Massachusetts Institute of Technology. http://senseable.mit.edu/co2go/images/co2go-technical-report.pdf.

Martin, B., Addona, V., Wolfson, J., Adomavicius, G., and Fan, Y. (2017). Methods for real-time prediction of the mode of travel using smartphone-based gps and accelerometer data. *Sensors*, 17(9).

Montini, L., Prost, S., Schrammel, J., Rieser-Schssler, N., and Axhausen, K. W. (2015). Comparison of travel diaries generated from smartphone data and dedicated gps devices. *Transportation Research Procedia*, 11:227 – 241. Transport Survey Methods: Embracing Behavioural and Technological Changes Selected contributions from the 10th International Conference on Transport Survey Methods 16-21 November 2014, Leura, Australia.

Montoya, D., Abiteboul, S., and Senellart, P. (2015). Hup-me: Inferring and reconciling a timeline of user activity from rich smartphone data. In *ACM SIGSPATIAL, International Conference on Advances in Geographic Information Systems*, pages 1–4, Seattle, WA, United States.

Newzoo (2017). Top 50 countries by smartphone users and penetration. https://newzoo.com/insights/rankings/top-50-countries-by-smartphone-penetration-and-users/ Visited 2/02/2018.

Nitsche, P., Widhalm, P., Breuss, S., and Maurer, P. (2012). A strategy on how to utilize smartphones for automatically reconstructing trips in travel surveys. *Procedia - Social and Behavioral Sciences*, 48:1033 – 1046. Transport Research Arena 2012.

Olah, C. (2015). Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ Visited 20/02/2018.

Reddy, S., Burke, J., Estrin, D., Hansen, M., and Srivastava, M. (2008). Determining transportation mode on mobile phones. In *2008 12th IEEE International Symposium on Wearable Computers*, pages 25–28.

Sauerlnder-Biebl, A., Brockfeld, E., Suske, D., and Melde, E. (2017). Evaluation of a transport mode detection using fuzzy rules. *Transportation Research Procedia*, 25:591 – 602. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.

Schüssler, N. and Axhausen, K. W. (2009). Processing gps raw data without additional information. *Transportation Research Record*, 2105:28 – 36.

Shafique, M. A. and Hato, E. (2015). Use of acceleration data for transportation mode prediction. *Transportation*, 42(1):163–188.

Shin, D., Aliaga, D., Tuner, B., Arisona, S. M., Kim, S., Znd, D., and Schmitt, G. (2015). Urban sensing: Using smartphones for transportation mode classification. *Computers,*

Environment and Urban Systems, 53:76 – 86. Special Issue on Volunteered Geographic Information.

Sonderen, T. (2016). Detection of transportation mode solely using smartphones. Technical report, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science. http://referaat.cs.utwente.nl/conference/25/paper/7555/detection-of-transportation-mode-solely-using-smartphones.pdf.

Stenneth, L., Wolfson, O., Yu, P. S., and Xu, B. (2011). Transportation mode detection using mobile phones and gis information. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 54–63, New York, NY, USA. ACM.

Vu, T. H., Dung, L., and Wang, J.-C. (2016). Transportation mode detection on mobile devices using recurrent nets. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 392–396, New York, NY, USA. ACM.

Wang, B., Gao, L., and Juan, Z. (2017). Travel mode detection using gps data and socioeconomic attributes based on a random forest classifier. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–12.

Widhalm, P., Nitsche, P., and Brndie, N. (2012). Transport mode detection with realistic smartphone sensor data. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 573–576.

Xiao, Y., Low, D., Bandara, T., Pathak, P., Lim, H. B., Goyal, D., Santos, J., Cottrill, C., Pereira, F., Zegras, C., and Ben-Akiva, M. (2012). Transportation activity analysis using smartphones. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 60–61.

Yu, M.-C., Yu, T., Wang, S.-C., Lin, C.-J., and Chang, E. Y. (2014). Big data small footprint: The design of a low-power classifier for detecting transportation modes. *PVLDB*, 7(13):1429–1440.