Traveler Mobility and Activity Pattern Inference Using Personal Smartphone Applications and Artificial Intelligence Methods

Ali Yazdizadeh

A Thesis

In the Department

of

Geography, Planning & Environmental Studies

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Geography, Urban and Environmental Studies) at

Concordia University

Montréal, Québec, Canada

May 2020

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ali Yazdizadeh

Entitled: Traveler Mobility and Activity Pattern Inference Using Personal Smartphone Applicationsand Artificial Intelligence Methods

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Geography, Urban and Environmental Studies))

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Gregory LeBlanc

_____ External Examiner
Dr. Francisco Camara Pereira

_____ External to Program
Dr. Ciprian Alecsandru

_____ Examiner
Dr. Nizar Bouguila

_____ Examiner
Dr. Gregory Butler

_____ Thesis Co-Supervisor
Dr. Zachary Patterson

_____ Thesis Co-Supervisor
Dr. Bilal Farooq

Approved by _____
Dr. Norma Rantisi, Graduate Program Director

March 12, 2020 _____
Dr. André Roy, Dean
Faculty of Arts and Science

Abstract

Traveler Mobility and Activity Pattern Inference Using Personal Smartphone Applications and Artificial Intelligence Methods

Ali Yazdizadeh, Ph.D.

Concordia University, 2020

Recent advances in communication technologies have enabled researchers to collect travel data from location-aware smartphones. These advances hold out the promise of allowing the automatic detection of the critical aspects (mode, purpose, etc.) of people's travel. This thesis investigates the application of artificial intelligence methods to infer mode of transport, trip purpose and transit itinerary from traveler trajectories gathered by smartphones. Supervised, Random Forest models are used to detect mode, purpose and transit itinerary of trips. Deep learning models, in particular, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), are also employed to infer mode of transport and trip purpose. The research also explores the use of Generative Adversarial Networks (GANs), as a semi-supervised learning approach, to classify trip mode. Moreover, we investigate the application of multi-task learning to simultaneously infer mode and purpose.

The research uses several different data sources. Trip trajectory data was collected by the MTL Trajet smartphone Travel Survey App, in 2016. Also, other complementary datasets, such as locational data from social media, land-use, General Transit Feed Specification (GTFS), and elevation data are exploited to infer trip information.

Mode of transport can be inferred with Random Forest models, ensemble CNN models, and RNN approaches with an accuracy of 87%, 91% and 86%, respectively. The Random Forest and multi-task RNN models to infer trip purpose achieve an accuracy of 71% and 78%, respectively. Also, the Random Forest transit itinerary inference model can predict used transit itineraries with an accuracy of 81%. While further improvement is required to enhance the performance of the developed artificial intelligence models on smartphone data, the results of the research indicate the

capability of smartphone-based travel surveys as a complementary (and potentially replacement) surveying tool to household travel surveys.

.

Acknowledgments

First and foremost I would like to express my gratitude and appreciation to my supervisors Zachary Patterson and Bilal Farooq. It has been an honor to be the first Ph.D. student of Zachary. Both Zachary and Bilal have taught me how good research in transportation planning and data analysis is done. I am grateful for all their vital contributions of time, ideas, encouragement and funding to make my Ph.D. experience creative and inspiring. They have been supportive and given me the freedom to pursue various approaches and extend my knowledge in machine learning and data analytics.

I also have to thank the members of my Ph.D. committee, Professor Francisco Camara Pereira, and Professor Ciprian Alecsandru for their beneficial suggestions and advice, as well as their time and consideration. Special thanks to the colleagues at TRIP Lab, Concordia, and LiTrans Lab, Ryerson: Kyle, Godwin, Mohsen, Marshall, Parham, David, and Arash, who helped me over past four years. My thanks also go out to the support I received from Concordia University, especially the Geography, Planning & Environment department, and Canada.

I would also like to say a heartfelt thanks to my wife, Nooshin, for inspiration and understanding, and my parents, brothers and sister, Mahnaz, Mahdi, Farnaz, Mohammad, and Hosein, for always supporting me and encouraging me to follow my dreams.

Contribution of Authors

Ali Yazdizadeh carried out the data processing, data analysis, and development of the machine learning algorithms in all of the four articles in this thesis. He also wrote the manuscript of the thesis. Prof. Zachary Patterson and Prof. Bilal Farooq supervised Ali Yazdizadeh during his PhD studies. They review all of the methods and manuscript text. All authors discussed the results and contributed to the final manuscript.

Contents

List of Figures

List of Tables

Glossary

Accuracy: The fraction of predictions that a classification model got right. In multi-class classification, accuracy is defined as follows [1]:

$$Accuracy = \frac{CorrectPredictions}{TotalNumberofExamples}$$

Activation Function: A mathematical function (such as sigmoid) that is fed by the sum of inputs, usually the inputs of the previous layer in a neural network, and then produces an (nonlinear) output value, to the next layer in a neural network [1].

Activity/Purpose: In this thesis an activity is considered as the purpose of trip at a destination, for example "Work" can be defined as an activity at trip destination. Activity and trip purpose have been used alternatively in this thesis.

Choice model: Choice model attempts to explain and predict the choice between two or more alternatives, usually mode of transport, such as taking bus or taxi to travel to work.

Choice set: Choice set is a set of available alternatives an individual is faced with while making a choice. The alternatives in a choice set should be mutually exclusive.

Classification model: A machine learning model for determining a label or class (among two or more classes) for an observation [1]. For example, a mode classification model can determine whether an input trip trajectory has been done by bike, car, walk or transit.

Computational graph: A directed graph consists of the nodes that are either operations or variables. The variable nodes contain multi-dimensional arrays (i.e. tensors) values. Operation nodes can be fed by the value of variable nodes or the output of other operation nodes. Indeed, every node in a computational graph defines a function of the variables [1].

Confusion matrix: A table that summarized the results of a classification model. It shows the correlation between the true label and the predicted label (i.e. the label generated by the model). Confusion Matrix usually consists of N rows and N columns, where N is the number of classes. The label predicted by the classification model is presented on one axis, and the true label is presented on the other axis [1].

F1-measure: The F1-measure is calculated as a weighted average of the precision and recall. It is defined by the following formula:

$$F1 - measure = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

Keras: An open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, or Theano [1].

Mode of transport: Mode defines the ways of transport and mobility, examples of modes of transport are car, bike, walk, etc.

Precision: Precision explains the frequency of the correct predicted positive class. It is defined by the following formula [1]:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Prediction: The output of a classification model while being fed by an input example [1].

Recall: Recall explains how many of the all true examples have been predicted correctly by the classification model. It is defined by the following formula [1]:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Segment: In this thesis a segment is considered as part of a trip's GPS trajectory with a specific number of GPS points.

Sequence model: A model which is fed by the sequential data, i.e. the data with sequential dependence. For example, a mode classification model which is fed by a GPS trajectory [1].

TensorFlow: A large-scale, distributed, machine learning platform. The term also refers to the base

API layer in the TensorFlow stack, which supports general computation on dataflow graphs [1].

Tour: A tour is a journey originated and destined at a same location (ex. work or home location) and consists of different trips with specified purposes and intervening stops.

Trajectory: A trajectory (or GPS trajectory) refers to a sequence of time-stamped points, usually recorded with some other information about latitude, longitude, altitude, speed, and acceleration, etc. The GPS trajectories can be recorded by different type of devices, such as GPS loggers or GPS-enabled smartphones.

Transit Itinerary: A transit itinerary is the details of scheduled events relating to a transit trip, generally including the bus number or metro line taken by the traveler at specific times and locations.

Trip: A trip is considered as a single journey between two points made by a specific mode of transport and has a defined purpose. As an example, a trip can be done from home to work by car.

Travel Time: While travel time is usually considered as the time it takes to travel door-to-door, in this thesis travel time is defined as the time between the first and last GPS point along a detected trip trajectory.

Chapter 1

Introduction

Contemporary cities, with rapidly changing urban mobility patterns resulting from the growth of new technologies, such as on-demand ride-sharing services like Uber and Lyft, location-aware smartphones as well as emerging technologies like autonomous vehicles, face transportation planning challenges. For several decades, transportation practitioners have used Household Travel Surveys (HTS), along with travel demand modeling [2], as a central tool in the analysis of transportation plans, projects, and policies in urban areas. Traditionally, HTSs have been done via various approaches such as face-to-face interviews, mail-back paper travel diaries, phone interviews (i.e. Computer-Assisted Telephone Interviews or CATI), or web-based travel surveys [3, 4].

With the rapid changes in technology, transportation planners need to be able to gather and update commuter datasets and models more frequently, for example on yearly basis, instead of every four or five years. However, traditional HTS methods suffer from several issues, such as heavy burden on respondents, high implementation costs, poor accuracy in terms of time and location, and lack of detail in recorded trips. Moreover, due to the high burden on respondents, response rates for such travel surveys tend to be low [3]. Furthermore, any attempts to improve the quality of collected data through the gathering of more detailed travel diaries increases respondent burden and negatively affects response rates further. In addition, the marginal cost[1] of implementing traditional HTS is usually relatively high [5], which leads to increasing implementation costs over time (as populations and sample sizes typically increase). Indeed, the high marginal cost of traditional HTS methods

[1] The additional cost of one more respondents.

not only increases their implementation costs over time, but also it may inhibit practitioners from expanding the study region easily or increasing their frequency. Hence, implementing HTS more frequently for growing urban areas requires confronting the above mentioned problems at the first step.

Over the last two decades, advances in other technologies, such as smartcards, the Global Positioning System (GPS), roadside sensors, and mobile phone data, have provided scientists and researchers with new and huge volume data sources, summarized under the term "Big Data". One of the advantages of such data gathering tools is that they collect data automatically with minimum need for human resources, which can decrease collection costs considerably. The use of such data sources has become an important avenue of research for understanding travel behavior. Particularly, smartphones equipped with different sensors such as Global Positioning Systems (GPS) and inertial measurement units (IMU), which comprise accelerometers and gyroscopes, have been changing the options available for the collection of transportation demand data. It should be mentioned that GPS loggers have also been used for gathering travelers trajectories, however such devices are not a practical tool in large-scale surveys, due to the relatively large burden and cost of their use. For example, Wolf et al. [6] used GPS loggers, carried by respondents, to collect traveler trajectory data. They report, however, that respondents often forgot to bring devices with them. Moreover, providing participants with dedicated GPS loggers increases surveying costs. Hence, the focus of this thesis is on smartphone travel surveys, which are different from surveys conducted via GPS loggers.

To take advantage of smartphone technologies in travel surveying, researchers have designed mobile applications to collect traveler mobility data [7, 8]. Travel surveys implemented via smartphone applications are one of the most suitable options to overcome the disadvantages of traditional HTS methods for the following reasons. First, they collect data automatically without imposing considerable burden on travelers. Second, their total cost is low compared to traditional HTS methods [5, 9]. For example, Flake et al. [9] have reported a 50% decrease in "cost per collected day" for their smartphone travel surveys over their household travel survey, both implemented in the same region. Third, the marginal cost of smartphone travel surveys is lower than traditional HTS methods. As Zhao et al [10] have found, the marginal cost of gathering an extra day of data with smartphone

travels surveys is minimal. Some studies [11] have estimated an increase between 23% to 67% in total surveying costs while adding an additional day in traditional HTS methods. However, adding one day to smartphone travel survey only increases the data administration expenses, which is minimal due to efficient cloud services. Today, the data administration systems of smartphone travel surveys can be implemented over cloud services, such as Amazon Web Services (AWS), which are cheaply scalable, and adding one day does not require large human resources. Such advantages of smartphone-based travel surveys will enable transportation practitioners to implement travel surveys more frequently at very low and reasonable costs, with lower respondent burden [12, 13, 14].

Nowadays, mobile phones equipped with different technologies can gather mobility and activity data. As shown in Figure 1.1, these technologies can be categorized into two types[15]:

- Locational technologies, such as cellular networks, GPS, WiFi, and Bluetooth,
- Built-in motion sensors, including accelerometer, magnetic sensors and gyroscopes.

The systems and technologies shown in Figure 1.1 are explained in Appendix A.1. Each of the above mentioned technologies possesses its own advantages and disadvantages, as explained in Appendix A.1. The most important issues are the accuracy and the accessibility of the data, as well as the effect of the data collection technique on mobile phone battery usage [16]. For example, cellular data, which is usually gathered by mobile phone network providers, is only obtainable through specific agreements and contracts, which potentially restricts their usage in transportation demand studies due to privacy issues [17]. Using more mobile phone sensors and technologies, such as accelerometer or gyroscope sensors along with embedded GPS technology, can potentially increase the data quality. However, deploying more sensors for data gathering may considerably increase smartphone battery usage, and may not be applicable to large-scale and long-term data collection surveys.

In this thesis, we used data from the application MTL Trajet, which is an instance of the smartphone travel survey app and platform, Itinerum [8, 18]. The Itinerum platform consists of a mobile application, which is installed on participant smartphones, and the web-based data administration platform. The mobile application collects the GPS records of travelers. It enables researchers to design a questionnaire inside the application to collect respondent socio-demographic and typical

travel details. It also prompts respondents when stops are detected, and asks them to validate trip mode and purpose. The models developed in thesis has been developed on MTL Trajet 2016 dataset, during which more than 8,000 individuals participated.

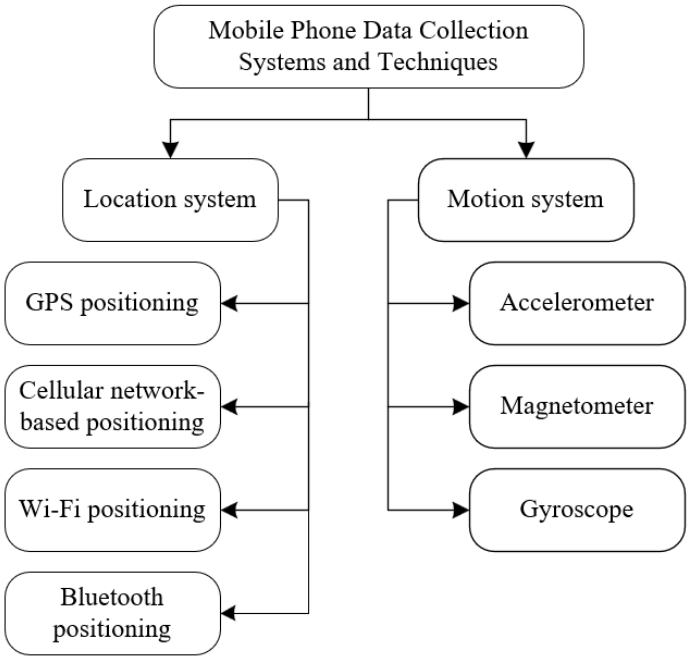


Figure 1.1: Different Mobile Phone Data Collection Technologies and Systems

While smartphone technologies promise great benefits for mobility data collection, there are many challenges in using them as a surveying tool in transportation demand studies. Such challenges are explained in the following section. Moreover, while smartphone-based travel surveys enable us to gather detailed travel data, GPS trajectories cannot be used directly in transportation demand analysis. Since the main aim for the use of travel data from smartphone surveys is for use in transportation demand analysis, we first need to consider the role of travel data in transportation demand modeling, and what type of data are essential in such uses. Hence, Section 1.2 is devoted to describing the role of travel data in transportation demand analysis and identifying the most important trip information in transportation demand modeling. Identifying the critical trip information in travel demand analysis helps us to determine which type of data should be collected by smartphone-based travel surveys and how we should analyse such data to elicit the information required in travel behavior studies. Following that, a section explains the different methods for inferring trip information from

smartphone-based travel surveys. Finally, the contribution of the thesis and dissertation outline are described.

1.1 Challenges in Using Mobile Phone Data in Transportation Studies

As explained in the previous section, smartphones have great potential to gather mobility data for transportation demand analysis. However, there are still some challenges to overcome. First, raw trajectory data collected by smartphones is not ready to use in transportation demand studies. For example, collected trajectory data cannot be directly converted into mode-based Origin-Destination (O-D) matrices, one of the the most important outputs of HTSs. To construct an OD matrix, we first need to be able to identify trips as well as trip mode and purpose. Such information needs to be inferred from raw collected GPS trajectories, before constructing an O-D matrix.

The second challenge is related to obtaining "Ground Truth" data for GPS trajectories[19]. Artificial intelligence researchers usually use the term "ground truthing" to refer to the series of actions or methods to achieve the proper data for testing the performance of AI algorithms[20]. With respect to trip trajectory data, ground truth data can include mode of transport associated to a trajectory by a respondent or a researcher. With ground truth data we can test the performance of inference algorithms and find out how much error may have occurred during the inference procedure. Obtaining ground truth data is not straightforward and usually is not achievable without placing some degree of burden on travelers. Different ground truthing methods are explained in Appendix A.2.

Third, many studies in the literature have been implemented in controlled environments where conditions are far from real-world situations. For example, as shown in Chapter 2, the majority of studies have been conducted on small-sized datasets or with small numbers of participants. Such limitations prevent researchers from generalizing their results for use with real-world populations. In other words, small-sized datasets are not always a broadly representative sample of the whole population, compared to large-scale household travel surveys. Moreover, survey data in many cases is gathered in controlled environments, i.e. with brand-new smartphones and without any concern about smartphone battery life. In large-scale and real-world implementations of smartphone-based

travel surveys, different groups of respondents, with different types of smartphones, may participate. While participants in research-based small-size surveys are probably more familiar with the travel survey app and more dedicated to collecting and validating data, due to the more involved training they usually receive, this may not be the case in the real-world implementations. Indeed, during large-scale travel surveys, it may not be possible to provide all the participants with the same level of training as those participants in small-size surveys, except when individuals are hired to train participants, which considerably increases implementation costs. Moreover, some participants may not use the application during the whole surveying interval, due to its negative impact on battery life. Furthermore, an application's ability to record data from all sensors may be restricted due to the differences between respondent smartphones. Hence, conducting the research on large-scale real-world datasets can provide a broader perspective on the challenges and limitations to replacing HTSs with smartphone-based travel surveys.

Fourth, there is the question of the degree to which smartphone users represent the larger population, and how much different age groups will participate in smartphone travel surveys. A particularly important concern is that older individuals, i.e. beyond 60 years old, may not be familiar with using smartphone applications, resulting in lower participation rates in smartphone-based travel surveys than younger age groups.

These issues currently present challenges to the use of smartphone travel surveys as a standalone travel surveying tool. However, as explained in the previous sections, their advantages, such as lower distribution and implementation costs, compared to traditional travel surveys, and not suffering from respondent fatigue or forgetfulness [21, 22], make them a suitable candidate for completely or largely replacing traditional household travel surveys. By using standalone smartphone travel surveys, transportation practitioners can benefit from a data collection technique that not only economizes their surveying costs, but also enables them to continuously gather traveler data over weeks or months, instead of a limited number of days. Also, they can enable transportation practitioners to collect data annually, instead of every four or five years, and from a huge number of individuals at a very low cost, compared to traditional HTS methods.

While transportation practitioners are faced with the aforementioned challenges, this thesis focuses on addressing the application of artificial intelligence to infer the most critical trip information

from mobile phone data. The results will help to enable practitioners to produce the information required in many travel demand studies from smartphone data. Moreover, the results can be used in transportation network design as explained in the following section.

Next section explains the role of travel diary data in transportation demand modeling. We cover different approaches in transportation behavioural analysis and identify their data requirements, which gives us an overview of main travel information that should be gathered by smartphone-based travel surveys.

1.2 The Role of Travel Diary Data in Transportation Demand Analysis

Information gathered by travel diaries, either via traditional surveying methods or smartphone-based travel surveys, is mainly used to model traveler behaviour, typically through trip-based, tour-based or activity-based models [2, 23]. Trip-based models are mainly focused on modeling the number of trips between origin and destinations, as well as trip- mode, purpose and route choice. The most well-known trip-based approach is the "four-step transportation demand model", which attempts to model the travel demand through four sequential steps: trip generation, trip distribution, mode choice and traffic assignment [2]. Activity-based travel demand modeling focuses on all possible combinations of activity and travels over the course of a day, instead of individual trips or tours [24]. Whether we use trip-based or activity-based demand modeling, we require travel information to develop our models. But how useful is data collected with smartphone-based travel diaries for transportation planners in their travel demand modeling studies? To answer this question, we need to know which trip information is used in transportation demand modeling.

In trip-based transportation demand analysis, the trip is the unit of travel or the unit of demand. For example, in trip-based demand modeling, the demand for transportation between two traffic zones is described by the number of trips originating from one zone and destined to another. Even, with an activity-based approach, each tour is comprised of several trips. A trip can be inferred by detecting its departure and arrival time. Hence, finding the departure and arrival time of the movement is the first step in processing the trajectory data, as it helps us to identify the trips.

Moreover, the output of the transportation demand studies is usually a mode-based origin-destination matrix, which is a matrix explaining how many trips by each mode of transport are originate from each traffic zone and are destined to other traffic zones. Also, in designing the road or transit network, knowing the share of each mode of transport is an important issue and helps transportation planners to properly allocate resources to develop transportation networks. Hence, mode of transport is another critical trip characteristic that should be inferred from smartphone trajectory data.

Furthermore, in transportation demand analysis, there is a widely-accepted principal that "*transportation is a derived demand*" [25], which means the need for travel is derived from another need, for example the need for work, education, recreation etc. More explicitly transportation demand is about satisfying a need for participating in an activity at the end of the trip, i.e. the destination. Hence, the purpose of trip at the destination plays an important role in many transportation demand studies and should be the focus of any study that aims to infer the trip information from smartphone trajectory data.

Furthermore, in travel demand studies, the last step of the "four-step transportation demand modeling", referred to as "Route Assignment", "Route choice", or "Traffic Assignment"[26] step. The traffic assignment step attempts to assign transportation demand to the transportation network, and focuses on modeling how travelers have chosen their route to move between their origins and destinations. Hence, we need to infer the chosen routes for each trip from trajectory data, as it is required in transportation demand modeling for traffic assignment step. However, for non-transit trips, such as trips taken by car, the road segments on which an individual has traveled can be inferred, with high accuracy, with the well-developed map-matching algorithms [27]. Hence, this thesis does not focus on inferring the routes for non-transit trip. However, finding transit routes from trajectory data is not straightforward and requires some considerations, as explained in Chapter 3.

Hence, transit itinerary (i.e. the bus route or metro line used for completing a transit trip) is another critical aspect of trip information that should be inferred from smartphone trajectory data. Also, from the perspective of transit network design, the demand for each transit route (i.e. metro line or bus route) is important information for transit scheduling and timetabling [28]. Moreover, transit route information, i.e. transit itinerary, has been collected via some traditional household travel surveys [29, 30, 31], and inferring it should be in the focus of any study attempting to show the

potential of smartphone-based travel surveys in transportation demand modeling.

To summarize, this thesis focuses on inferring the most important trip features that are typically used in transportation demand studies and forecasting, i.e.:

(1) Trips (departure and arrival time and location)

(2) Mode of transport

(3) Purpose of the trip

(4) Transit itinerary (the bus route or metro line used for completing a transit trip)

These attributes are the basic characteristics of a trip, upon which we can infer other travel characteristics. For example, the "travel cost" or "out-of-vehicle travel time" can be inferred using traditional techniques upon detecting the mode of transport and departure and arrival time. Even trip tours, used in activity-based demand analysis, can be constructed by joining together the successive trips. Among the aforementioned, this thesis concentrates on the characteristics 2-4. While trip determination is critical for the rest of the analysis, we adopt a typical rule-based approach, which demonstrated good results in detecting trips from GPS trajectories collected by the Itinerum platform. The details of the algorithm are explained in Chapter3.

To infer the other important trip information, i.e. mode of transport, trip purpose, and transit itinerary, this study focuses on the application of Artificial Intelligence (AI) methods on smartphone-based travel survey data. In the next section, the inference methods in artificial intelligence domain are briefly reviewed and the most appropriate ones applicable for trajectory data are explained. In Section 2.1, we explain why the different artificial intelligence methods are appropriate for inferring trip information.

1.3 Inference Methods

This section introduces the inference methods applicable to smartphone data to infer trip mode, purpose and transit trip itinerary. The most basic inference methods to analyze smartphone trajectory data are rule-based methods, which try to predict mode or purpose or detect the stop and transition

points, based on a set of formulated rules, established upon the experience and expertise of scientists and researchers in the field. However, methods developed in other fields of science, such as statistics, computer vision and natural language processing, have the potential to improve the accuracy of mode, purpose or transit itinerary prediction from smartphone-based travel data and have been used in the literature. In this thesis three terms have been used, sometimes alternatively, to address such inference methods: "Artificial Intelligence", "Machine Learning", and "Deep Learning". The relationship between these three terms is shown in Figure 1.2. Artificial intelligence was first introduced in 1956 [32] and refers to automated inference systems to emulate human inference [33]. Machine learning models are a subset of artificial intelligence and learn from the data which is provided to them. By "learning" we refer to the procedure consisting of estimating model parameters so that the learned model (algorithm) can perform a specific task [34].

Deep Learning is a subset of machine learning, which deals with learning via neural networks. While the concepts behind neural network algorithms have been developed since 1980s [35], the popularity of the term "deep learning" increased with industrial applications of neural networks since the late 2000s, especially those using Convolutional Neural Networks (CNN) in image recognition and Recurrent Neural Networks in speech recognition tasks [34]. The word "deep" refers to the number of layers in neural networks, i.e. the higher the number of layers, the deeper the neural network.

In this thesis, the term "artificial intelligence" is used to refer to inference methods in general. The term "machine learning" addresses the classical inference algorithms, such as decision trees, Support Vector Machines or Bayesian Networks, which are able to learn from data. The term "deep learning" refers to neural network algorithms developed in natural language processing or image recognition, such as Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) [33].

The machine learning algorithms applicable to travel trajectory data can be categorized based on three criteria, as shown in Figure 1.4:

- Label type: Labeled and unlabeled data
- Feature Type: disaggregated point-based or aggregated segment-based features

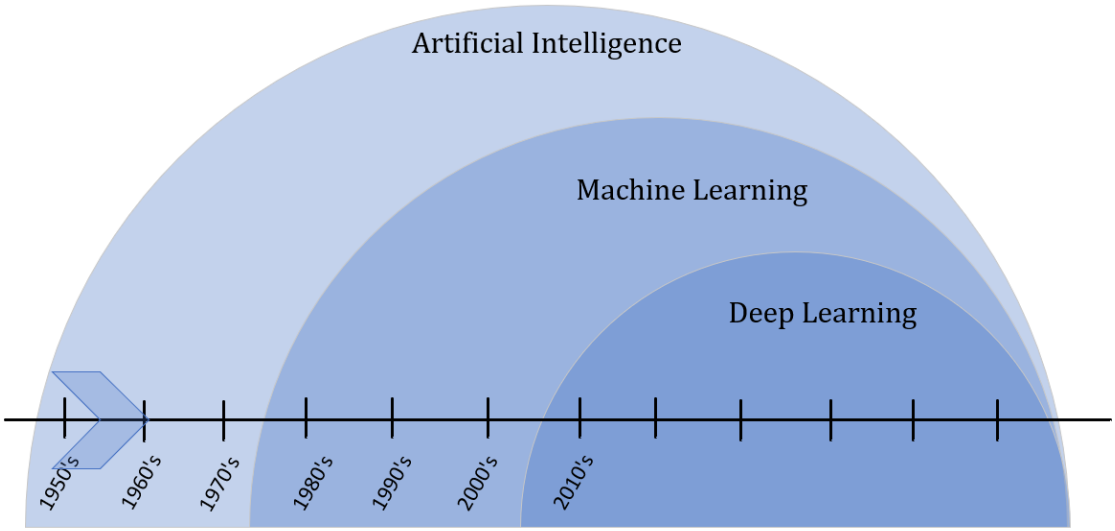


Figure 1.2: Artificial Intelligence VS. Machine Learning VS. Deep Learning

- Number of learned tasks (labels) per each observation: Single-task and Multi-task learning

We explain each of these categories below. First, with respect to label type, there are three types of learning approaches in artificial intelligence research: supervised, semi-supervised and unsupervised learning [33, 36], based on whether input data is labeled or unlabeled. In supervised learning, a set of labeled training data is used to infer a function that maps an input to an output based on input-output pairs. Semi-supervised learning is also a supervised learning class but where only a subset of training data is labeled. Semi-supervised learning methods can still take advantage of unlabeled data for training. In unsupervised learning, all training data are unlabeled. Unsupervised learning infers a function that describes the structure of data and groups them. Supervised learning approaches are highly dependent on labeled data, i.e. the data for which the class or label of each observation was previously validated by a human. For example, the mode of transport of each trip can be validated by travelers in the travel survey smartphone application upon detecting a stop or reaching a destination. On the other hand, it is often the case that not all data in a dataset (collected by a smartphone-based travel survey) has labels, if e.g. respondents haven't validated mode of transport for a trip. Semi-supervised learning approaches try to estimate a label for unlabeled data and use them along with labeled data for data classification [2] [37]. Also, semi-supervised generative

[2]The definition of the terms in artificial intelligence or transportation planning domain has been provided in

models, such as generative adversarial networks, can generate observations, i.e. the characteristics of observation are generated, by looking at the statistical distribution of the real labeled data. Unsupervised learners are those applied on completely unlabeled data for categorizing data into a pre-defined number of categories [33]. Each of these learner types have their advantages and disadvantages. However, the majority of learners developed in different fields of science have tended to use supervised learning. Semi-supervised and unsupervised learners are the approaches most open to further investigation.

Second, based on our knowledge and experience during doing this thesis, classification methods for analyzing trip trajectory data can be classified into segment-based and point-based methods, based on whether the features used in the model are aggregate or disaggregate features. As shown in Figure1.3, in point-based methods, the features of each GPS point are analyzed in the prediction procedure at a disaggregate level. In the segment-based methods, aggregated features for the whole trip trajectory, such as average speed, are calculated and used in prediction procedure.

For example, for mode of transport classification, point-based classification methods try to predict mode of transport based on a set of features for "each point" along a trip, ex. longitude, latitude and time of each GPS point as shown in Figure 1.3. Segment-based methods, on the other hand, predict mode based on aggregate features of the whole trip, such as average speed, as shown in Figure 1.3. Point-based methods should be capable of being fed all the points along a trip and considering the features of each point in their prediction process. The example of such models are the ones developed in image recognition or natural language processing. In image recognition, RGB values, i.e. the values for Red, Green and Blue colors, of all the pixels in an image are fed to a neural network model. In language processing, the words in a sentence comprise a sequential data form, which is fed into a neural network. Similarly, each point along a trip can play the role of a pixel in an image or the role of a single word in a phrase. Considering the points as pixels or words enables the researchers to use the pioneering algorithms in machine/deep learning research.

Third, artificial intelligence algorithms can be categorized into two groups based on the number of tasks they infer[38]: single-task and multi-task classifiers. Single-task classifiers predict only one task at a time, for example, a single-task mode inference model is only able to predict the mode of transport for each trip trajectory. On the other hand, there are multi-task learners, which deal

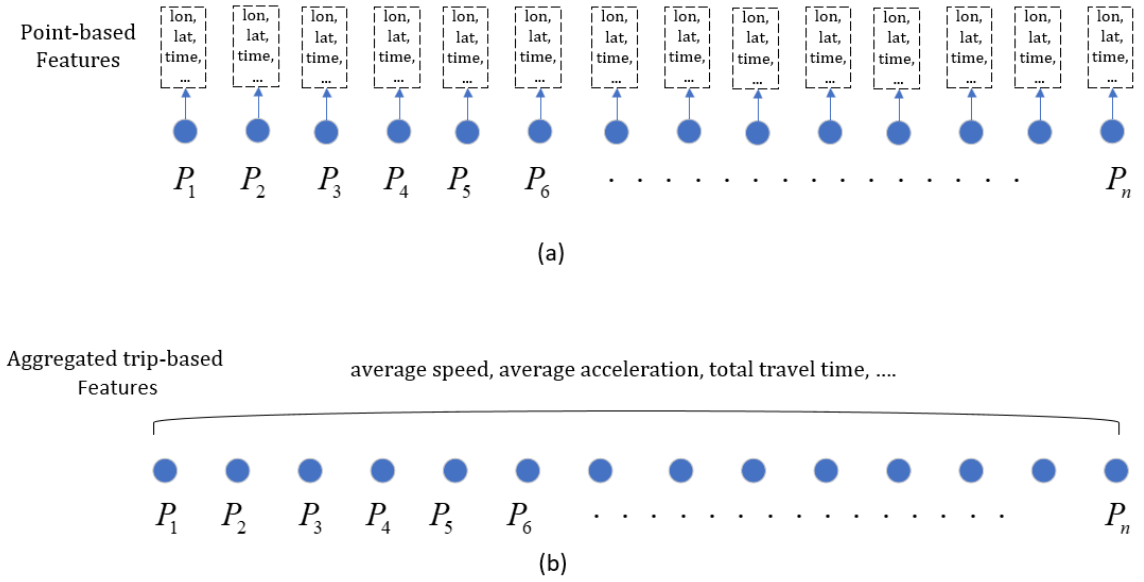


Figure 1.3: (a) Disaggregated Points-based Features VS. (b) Aggregated segment-based Attributes ("lon" means longitude, "lat" means latitude)

with two or more tasks simultaneously. As an example, a multi-task classifier to predict mode and purpose outputs two tasks simultaneously. Multi-task learners take advantage of situations in which two tasks are strongly correlated, like mode of transport and trip purpose in transportation planning studies.

In this thesis, we start by developing Random Forest models in Chapter 3, as supervised single-task segment-based learners to infer the important trip information, i.e. the mode of transport, purpose of trip and transit itinerary. Afterwards, Chapter 4 examines the application of Convolutional Neural Networks (CNN) as a point-based learner to take advantage of the detailed information related to each GPS point along a trip in the learning procedure. Next, to test the application of a semi-supervised technique, Generative Adversarial Networks (GANs) on point-based features are implemented. Both the CNN and GANs models in this thesis are single-task learners. Finally, in Chapter 6, we examine supervised multi-task learners on both point-based and segment-based features, using two Recurrent Neural Network (RNN) approaches, i.e. Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU) and compare their prediction results with single-task learners.

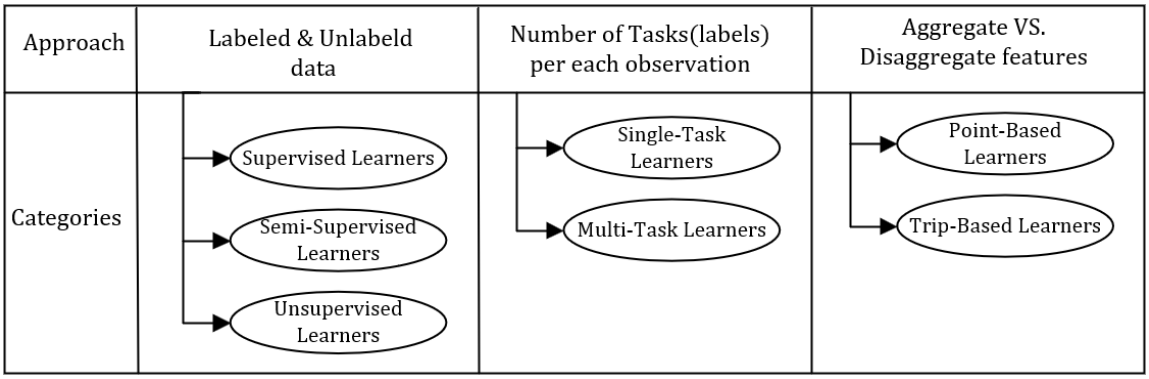


Figure 1.4: Categorization of Artificial Intelligence Algorithms for Analyzing Smartphone-based Travel Data

1.4 Contribution

The research in this dissertation contributes to the academic literature in the following ways:

Trip information inference: Recent supervised methodologies and frameworks from computer vision and natural language processing are investigated using Convolutional Neural Networks and Recurrent Neural Networks to infer mode of transport. Furthermore, a semi-supervised modeling approach is examined using generative adversarial networks to detect mode of transport. Also, the effectiveness of ensemble methods in inferring transportation mode is tested and their superiority over single learners is shown. The state of the art in deep learning studies is explored to infer the mode of transport from trajectory data.

For trip purpose, both machine learning and deep learning models, i.e. Recurrent Neural Networks and variants, are applied on trajectory data to infer trip purpose. The deep learning approach uses both point-based characteristics and socio-demographics, as auxiliary data, to infer trip purpose.

With respect to transit itinerary, a machine learning framework is developed to predict the corresponding bus routes and metro lines of trajectory data.

Overall, the above models and frameworks are developed on the MTL Trajet/Itinerum dataset to show the high potential in real-world smartphone-based travel survey to replace the traditional household travel surveys.

Inclusion of Complementary Data sources in inferring trip information: In addition to the main MTL Trajet dataset, a variety of data sources were used as complementary data to improve the

prediction accuracy of inference models. Social media data, Google Transit Feed Specification (GTFS) data, elevation data, and land use data are fed into different modeling approaches to enhance the prediction accuracy of trip information.

Multi-task learning of mode and purpose: To take advantage of the correlation between mode and purpose of each trip, a multi-task learning approach is developed based on different types of Recurrent Neural Networks. The model depicts how jointly predicting mode and purpose can improve the prediction accuracy of each, compared with a single-task learning approach.

1.5 Dissertation Outline

The main contributions of the dissertation are divided into four articles/chapters, each focusing on different methodological approaches to infer trip information from smartphone-based travel surveys. Before that, a chapter is dedicated to reviewing the literature and explain the background of the models and approaches used in the study. The detailed formulas and mathematical operations of Random Forest models, Convolutional Neural Networks, Recurrent Neural Networks, and generative adversarial networks are explained in Chapter 2.

Chapter 3, introduces the Random Forest models developed to detect all trip information from trajectory data, i.e. mode, purpose, and transit itinerary. A rich source of complementary data and their importance in the prediction of the trip information has been analyzed and explained in Chapter 3.

The subsequent chapter, Chapter 4, is devoted to Convolutional Neural Networks and ensemble methods in inferring mode of transport. Different point-based attributes for each trajectory are fed to the developed Convolutional Neural Networks to infer mode of transport. Also, different ensemble methods are explained and their performance tested against each other.

Chapter 5 concentrates on the application of semi-supervised Generative Adversarial Networks (GANs) to infer mode of transport. A GANs classifier is developed using Convolutional Neural Networks as discriminator and generator. The performance of the GANs model is tested in this chapter.

The subsequent chapter, Chapter 6, deals with multi-task learners and shows their superiority over

single-task models. Different single-task Gated Recurrent Units (GRU) and Long-Short Term Memory (LSTM) are developed and their performance is compared with each other. Also, the trajectory data are fed into multi-task learner frameworks along with auxiliary data, i.e. socio-demographics and destination-related data, to predict the mode of transport and purpose of trip simultaneously.

Finally, the concluding chapter, Chapter 7, summarizes the findings of this thesis and provides directions for future studies.

Chapter 2

Background

This chapter reviews the literature of inference methods on mobile phone data to infer trip information. It also introduces the background of the models used in this thesis. The current chapter comprises two subsections: first, a review on mode, purpose and transit itinerary studies. The different methods are categorized and reviewed based on criteria in Figure 1.4. The next section, i.e. Section 2.2, covers the background of models developed in this thesis. All the models explained in Section 2.2, are developed in the field of artificial intelligence, mainly in image classification and natural language processing (NLP). We explain each model in detail.

2.1 Literature Review

This section reviews the studies on inferring trip information, i.e. mode of transport, purpose of trip, and transit itinerary from mobile phone data. It then attempts to review different studies and categorize them based on the criteria demonstrated in Figure 1.4. Also, dataset size and validation methods used in each study are reviewed. The goal of this section is to explain which categories (in Figure 1.4) have been covered in the literature, and which categories are worth of more attention. This section starts with reviewing studies on mode, purpose and transit itinerary inference. Finally, the section concludes on different aspects of the literature and shows the directions that researchers should address in future studies.

2.1.1 Mode Detection

The literature on mode detection is larger than trip purpose or transit itinerary. Several researchers have proposed algorithms to infer mode of transport from various smartphone data sensors, such as mobile phone GPS and accelerometer data [39, 40], or mobile phone cellular data [41]. While some studies have used only one data source, such as GPS data [42], others [43] have used combinations of GPS and accelerometer data to infer mode of transport. Furthermore, some studies have used other mobile sensors, such as magnetometers or gyroscopes, to gather mobility data to infer mode of transport [44, 45]. While there are several systems and technologies embedded in smartphones, as explained in Appendix A.1, using all of these technologies is not feasible and practical in large scale data collection efforts, as not all respondent mobile phones are equipped with the all types of sensors and technologies. As such, and as is clear from Table 2.1, the majority of research has been conducted using GPS and accelerometer data.

The methods in the mode detection literature have adopted various approaches, including: rule-based [46, 47], fuzzy logic [48, 49], artificial intelligence [39, 40], and discrete choice[50] approaches. As shown in Table 2.1, the studies in the literature have been primarily done using the aggregate features for whole trips, or segments of the trip, instead of analyzing detailed information of each GPS point. Furthermore, a few studies [37, 51] have investigated the application of semi-supervised learning approaches.

With respect to dataset size and groundtruthing procedures (Different ground truth procedures are explained in Appendix A.2), the majority of datasets reported in the literature are small in size or have been collected by only a small number of participants, as shown in fifth column of Table 2.1. The large difference between dataset sizes for studies reported in the literature compared with what real-world HTS studies has a major implication: the majority of experiments in the literature have been done with small number of smartphones, where data or labelling quality is potentially far from a real-world experiment. For example, the US National Household Travel Survey [52], conducted on 2009, collected data from more than 150,000 households. In Montreal, the latest CATI Origin-Destination survey surveyed around 74,000 households [53].

In a real-world smartphone-based survey, there are large numbers of participants with many different types of smartphones, each equipped with different sensors and capabilities. For example, in MTL Trajet travel survey [18] around 8,000 individual participated. Moreover, in small-size experiments, participants have usually received training to use the surveying app and validate trip information. In real-world experiments it may not always be always possible to train participants as thoroughly. Furthermore, labeling procedures, that produce ground truth data, vary among studies: from prompted recall surveys [7, 54] and in-app validation [39] to manually annotated surveys [55]. Prompted recall surveys potentially can collect more accurate and detailed data. However, as they increase surveying implementation costs and place more burden on respondents, there is always a trade-off between collecting higher-quality data and increasing the surveying costs.

Besides the aforementioned differences among studies, in the following, the studies on mode detection, and their methodology and results are explained. Tree-based ensemble classification algorithms have been used by Xiao et al. [40] to classify mode of transport. Their best ensemble method achieved a prediction accuracy of 90.77%. Wang et al. [56] develop a Random Forest classifier combined with a rule-based method to detect six modes of transportation using seven GPS-related features. Their method is able to detect more than 98% of subway trips with an overall accuracy of the classification of the other five modes being as high as 93.11%. Assemi et al. [57] deployed a nested logit model with eight attributes to infer mode of transport from smartphone travel surveys, implemented in New Zealand and Australia. They have reported an accuracy of 97% for New Zealand which includes data cleaning and 79.3% for Australia without any pre-processing.

Endo et al. [58] proposed a deep neural network approach to automatically extract high-level features. Their innovative approach converted raw GPS trajectories into a 2-D image structure and fed it as the input to a deep neural network. As an alternative to the RGB (Red, Green, Blue) values of an image pixel, stay time, i.e. the duration that a user stays in the location of the pixel, was used as the pixel value. They integrated hand-crafted features with image-based features. Eventually, they deployed traditional machine learning models, such as logistic regression and decision tree, to predict mode of transport. Although they devised an innovative idea to convert GPS trajectories into 2-D images, the pixel values only contained stay time without taking into account the spatiotemporal or motion characteristics, such as speed or acceleration, of the GPS trajectories.

Table 2.1: Mode Detection Models in the Literature

Model	Categorization	Data Type	Features Type	Dataset Size	Ground Truth
Neural Networks (Gonzales et al. [59])	Supervised, Single-task	GPS Positioning	Aggregated	114 trips	Manually annotated
Neural Network (Yang et al. [60])	Supervised, Single-task	GPS positioning	Aggregated	20 participants	Unknown
Neural Networks (Byon and Liang [61])	Supervised, Single-task	GPS positioning, accelerometer, magnetometer	Aggregated	5 participants	Unknown
Neural Networks and Particle Swarm Optimization (Xiao et al.[54])	Supervised, Single-task	GPS positioning	Aggregated	Unknown	Prompted recall survey
Bayesian Networks (Xiao et al.[62])	Supervised, Single-task	GPS positioning	Aggregated	Unknown	Prompted recall survey.
Random Forest (Lari and Golroo[63])	Supervised, Single-task	GPS positioning, accelerometer	Aggregated	35 participants	Manually annotated
Random Forest (RF), Naïve Bayesian (NB), and Multilayer Perceptron (ML) (Stenneth et al.[42])	Supervised, Single-task	GPS positioning	Aggregated	6 participants	in-app validation
Support Vector Machine (SVM) and Conditional Random Field (CRF) (Zheng et al.[64])	Supervised, Single-task	GPS positioning	Aggregated	45 participants	Prompted recall survey
An ensemble of probabilistic classifiers combined with a Hidden Markov Model(Nitsche et al.[55])	Supervised, Single-task	GPS positioning	Aggregated	15 participants	Manually annotated
Support Vector Machines (Zhang et al[65])	Supervised, Single-task	GPS positioning	Aggregated	197 trajectories	in-app validation and modification
Decision Tree and discrete Hidden Markov Model (Reddy et al.[66])	Supervised, Single-task	GPS positioning, accelerometer	Aggregated	16 participants	Experiment (i.e., mode known)
Tree-Based Ensemble Classifiers (Xiao et al.[40])	Supervised, Single-task	GPS positioning	Aggregated	73 participants	in-app validation
Bayesian Belief Network (Feng&Timmermans[43]	Supervised, Single-task	Accelerometer enabled GPS Device	Aggregated	Small sample (not exactly stated)	Manually annotated
Hierarchical classification (Support Vector Machine and Decision Tree) (Soares et al.[67])	Supervised, Single-task	GPS positioning	Aggregated	19 participants	Manually annotated
Logit models (Broach et al.[68])	Supervised, Single-task	GPS positioning, accelerometer	Aggregated	80 participants	Recall validation through mail
Convolutional Neural Networks (Dabiri$Heaslip [39])	Supervised, Single-task	GPS positioning	Disaggregated	69 participants	in-app validation
Convolutional Autoencoder(Dabiri$Heaslip [51])	Semi-Supervised, Single-task	GPS positioning	Disaggregated	69 participants	in-app validation
Label propagation with KNN kernel (Rezaie et al.[37])	Semi-Supervised, Single-Task	GPS positioning,	Aggregated	702 trips	in-app validation

Their best models were able to detect the mode of transport with prediction accuracy of 67.9% on the GeoLife dataset and 83.2% on the Kanto Trajectories dataset. Zhang et al. [65] suggested a multi-stage method to identify mode of transport. In the first stage they used a rule-based algorithm using three features, mean speed, maximum speed and heading rate, to distinguish between motorized, i.e. walking and cycling, and non-motorized modes of transport. In the next stage, they proposed a Support Vector Machine (SVM) to classify car, bus, tram and train. They reported a high prediction accuracy rate of 93%. They have used an in-app validation and modification procedure, where travelers can specify the start and stop of each trip as well as selecting their mode of transport from a list. Moreover, their application gives travelers the chance to modify trip information later. Furthermore, the application sends users notifications and wants them to identify their mode of transport every 10 minutes. Also, whenever the application detects a signal loss more than 20 seconds, it sends notifications after gaining the signal and ask the users to validate their mode of transport, to record the possible change in mode of transport during the signal loss. Such consideration enabled the researcher to gather more accurate data and prevent the signal loss effect on labeling quality.

Regarding point-based algorithms, which are able to be fed by disaggregate GPS point features, more recently, Dabiri and Heaslip [39] used CNN models to train a mode detection classifier. They developed different architectures of CNN models on GPS trajectories, and finally combined their output via an ensemble method. Their ensemble library comprised seven CNN models. They took the average of the softmax class probabilities, predicted by each CNN model to generate the transportation label posteriors.

With respect to semi-supervised algorithms, Rezaie et al. [37] have used a semi-supervised approach to infer mode of transport from GPS trajectory data. They implemented the Label Propagation (LP) technique using the k-nearest neighbors (KNN) algorithm to produce labels for the unlabeled part of the data to find out how much it may enhance the prediction accuracy rate of the mode classification. They compared the performance of semi-supervised LP models against Random Forest (RF) and Decision Tree (DT) classifiers. They found that the RF model in many cases performs better than the semi-supervised LP approach to infer mod of transport from GPS trajectory. Their results demonstrates that only where the proportion of unlabeled data is more than 80% of the

training data, the LP model outperforms the RF model. For example they reported 83% and 81% prediction accuracy for LP and RF model respectively, where the training dataset comprised of 80% unlabeled data.

In the context of semi-supervised learning for mode of transport inference, Dabiri et al. [51] have developed Variational Auto-Encoders to detect mode of transport from GPS trajectories using both labeled and unlabeled data in the dataset. Their proposed Semi-supervised Variational Auto-Encoder model outperformed the supervised models, such as Convolutional Neural Networks and Recurrent Neural Networks. Specifically, when they included all the labeled segments in the training dataset, their proposed semi-supervised model achieved a prediction accuracy of 76.8%, where their best supervised approach achieved an accuracy of 71.4%.

This section reviewed the studies on detecting mode of transport from GPS trajectory data. The rule-based and supervised machine learning algorithms have been well covered in the literature. However, the semi-supervised algorithms and deep learning model have not been thoroughly investigated by the researcher. Moreover, the majority of studies have been carried out on small sized datasets, which prevent transportation practitioners from generalizing the results.

2.1.2 Purpose Detection

The literature on trip purpose detection from mobile phone data is not as extensive as mode inference. This section reviews purpose detection methods applied on GPS, smartphone and HTS data. Several approaches have been utilized to detect trip purpose from mobile phone data: rule-based algorithms, probabilistic methods and artificial intelligence methods [69]. Rule-based methods in the field of trip-purpose detection are the most common ones [70]. Rule-based methods involve a set of heuristic rules to infer trip purpose from GPS data, and other data sources, such as land-use data. Wolf et al. [71] developed a set of rules to find the correspondence between the land-use data and trip purpose. They combined these rules with dwell time and occupation to determine trip purpose.

However, despite their popularity in trip purpose identification literature, the rule-based algorithms suffer from several limitations: first, they depend on background knowledge of the model builder [72]. Second, they depend on the dataset that they are built on. For example, the rules to infer trip

purpose need to be modified depending on the resolution of land-use data, as in some regions the land-use data is available in parcel-level, while in other regions we may access only to the land-use information in lower resolutions, such as census divisions or blocks.

The second type of trip purpose inference methods are probabilistic methods. Axhausen et al. [73] established a set of matching rules to impute the purpose of trip using data gathered by GPS devices installed on 186 vehicles. They clustered destination locations based on the closeness ($<$200m) to the driver's household location. Activities in the vicinity of home location were assigned as *Home* or *Return Home* purpose. For other trip purposes, they used Point of Interest (POI) information such as restaurants, gas stations, as well as land-use data to impute trip purpose. They defined a distance of 300-meter around each trip destination to define an evaluation area of potential activities. Afterwards, they calculated the probability of each type of activity in the defined area around each destination. They also assigned a weight to each activity type with respect to its distance to the destination. For example, activities within 50m of destinations were assigned the weight of 1.5; between 50m and 100m the weight of 1, and so on. They do the similar procedure for land-use data. Finally, they assigned trip purpose to each trip destination based on calculated activities in the destination vicinity and a set of rules. It should be mentioned that their data was not labeled and they compared their results with the national household travel survey to find how similar imputed trip purpose was to the reported purpose of similar trips in the household travel survey. While the probabilistic methods infers trip purpose based on the calculated probability of surrounding activities around a destination, such methods still rely on a set of decision rules to estimate trip purpose from the calculated probabilities.

Artificial intelligence methods have been rarely applied to estimate trip purpose [70]. McGowen et al. [74] used decision tree models to infer trip purpose from GPS trajectory collected by GPS loggers attached to each traveler. They used land-use and socio-demographic data as complementary data sources. They have also applied a discriminant analysis to infer trip purpose. Deng et al. [75] used decision tree models to infer trip purpose using trip ending time, mode of transport, travel time and length, and socio-demographics such as age, income, etc.

Gong et al. [70] developed several machine learning approaches, including a classification tree, Support Vector Machine, neural network, and discriminant analysis methods. Their results show the

superiority of tree-based methods over the other tested algorithms. They categorized the features into three categories: activity features, trip features, weather features, and demographic features; all of which aggregated features or features related to the destination point or socio-demographics.

Oliveira et al. [76] developed a decision tree model to differentiate between 12 trip purposes. In their study, a decision tree model achieved a prediction accuracy of 65%. They also used a Nested Logit model that predicted trip purposes with 60% accuracy. Wu et al. [4] developed two models, a rule-based and a Random Forest model to classify indoor, outdoor static (i.e., when an individual is relatively stationary while outdoors), outdoor walking, and in-vehicle travel activities. Both of the models successfully predicted indoor activities and in-vehicle travel with 96% and 88% accuracy, respectively. However, they observed moderate performance for identifying walking trips, compared to the performance of their methods in identifying the other modes of transport. Also, they have reported a low precision (17.6 %) for detecting outdoor statistics. Furthermore, they did not find any considerable differences in performance between the rule-based and the Random Forest models. They concluded that the Random Forest model was easy to implement and efficient in terms of running time, but showed less robustness than the rule-based model while being implemented on biased or poor quality training data. Kim et al. [77] developed a Random Forest model to differentiate between 16 purposes. They used age and gender variables in the modeling process to improve the prediction performance of the model. Their Random Forest model can predict trip purpose by 75.5% prediction accuracy.

Table 2.2: Trip Purpose Detection Models in the Literature

Model	**Categorization**	**Data Type**	**Features**	**Dataset Size**	**Ground Truth**
Multi-stage rule-based approach (Axhausen et al. [73])	Imputation, Single-task	GPS devices on vehicles	Aggregated	186 participants	Manually annotated
Decision Tree (Deng&Ji [75])	Supervised, Single-task	GPS device attached to travelers	Aggregated	226 participants	Manually annotated
Decision Tree (Mcgowen& McNally. [74])	Supervised, Single-task	GPS devices attached to travelers	Aggregated,	36 participants	Annotated on a web-based questionaire
Decision tree, Support Vector Machine, neural network (Gong et al.[70])	Supervised, Single-task	GPS devices attached to travelers	Aggregated	20 participants	Travel plans entered before the first trip of the day
Rule-based (Stopher [46])	Supervised, Single-task	GPS devices attached to travelers	Aggregated	58 households (137 participated)	Manually annotated on paper
Random Forest (Wu et al.[4])	Supervised, Single-task	GPS devices attached to travelers	Aggregated	3 participants	follow-up interviews
Support Vector Machine (Zhu et al.[78])	Supervised, Single-task	HTS survey(No GPS data)	Aggregated	10,372 participants	Unknown
Random Forest (Kim et al.[77])	Supervised, Single-task	GPS positioning on smartphones	Aggregated	nearly 1000 participants	prompted-recall travel survey
Random Forest, Nested Logit (Ermagun et al.[79])	Supervised, Single-task	HTS survey(No GPS data)	Aggregated	30,286 persons	Unknown
Rule-based (Fang et al.[80])	Supervised, Single-task	GPS positioning	Aggregated	950 respondents	Manually annotated on paper

Support Vector Machines (SVM) were used by Zhu et al. [78] to infer trip purpose from household travel survey data. They used location-based social network data to augment household travel survey data. They have demographic features, temporal features, like the duration stayed in a destination, and other spatial features, such as the nearby venue from Foursquare [81] data. Their SVM model classified trip purpose with 75.28%.

Kim et al. [77] implemented ensemble learning to infer trip purpose from the Future Mobility Survey (FMS) [7]. The dataset is among the largest datasets, nearly 1,000 participants, collected via a prompted recall smartphone-based survey. The developed Random Forest model predicts 16 activity classes with 75.54% accuracy. They also tested the performance of a Random Forest model to classify 4 activity classes, instead of 16 classes, and found higher prediction performance, i.e. 84.08%, in trip purpose classification.

Some trip purpose inference studies have been implemented on large-scale traditional household travel surveys. Ermagun et al. [79] deployed Random Forest and Nested Logit models to infer trip purpose for trip destination data collected during "2010 Travel Behavior Inventory (TBI)" survey, in the "Twin Cities". The dataset comprised a travel diary from 30,286 individuals [79]. Random Forest and Nested Logit models predicted trip purpose with 57.69% and 64.17% prediction accuracy, respectively. They used Google Places locational data as complementary data to HTS data. In a similar context, Zhu et al. [78] investigated the potential of location-based social networks to predict trip purpose based on data from traditional household travel surveys. They trained a Support Vector Machine (SVM) algorithm that achieved over 75% accuracy in predicting trip purposes.

This section reviewed the studies on trip purpose detection from GPS, mobile phone and HTS data. Although the literature on trip purpose is not as rich as the literature on mode detection, rule-based algorithms and some machine learning models have covered in the literature. However, as demonstrated in Table 2.2, the majority of models to detect trip purpose have been developed on small-size datasets obtained from GPS devices or mobile phone data. Hence, more investigation is required on developing machine/deep learning methods on larger dataset sizes to detect trip purpose. The next section addresses the limitations in the current literature on mode and purpose detection. Since the literature on transit itinerary inference is too small, we mention it briefly in the following section as well.

2.1.3 Limitations of Existing Work and Future Prospects

In the previous sections, the literature on inference methods on mobile phone data to elicit trip information was reviewed. With respect to mode of transport, As shown in Table 2.1- 2.2 the majority of the literature is based on aggregate trip(segment)-based features, while fewer focus on disaggregate point-based features, to infer mode of transport. That most work has been based on aggregate data is partially due to the characteristics of classical machine learning models, like the Random Forest or SVMs, which cannot be fed the detailed features of each GPS point along a trajectory or each pixel in an image. However, the rise of deep learning models, specifically the CNN and RNN models, have recently provided researchers with algorithms able to process detailed features of each point (e.g. along a trajectory), or each pixel (in an image), or each word (in a sentence). Given the success that such dissagregate methods have had in other fields of computer science (e.g. image recognition), it is now possible to evaluate their potential in the context of disaggregate spatial data from smartphones. Given the limited use of such methods in transportation as well as increasing availability of disaggregate data, there is also a need to evaluate such algorithms and examine their advantages and disadvantages in mode of transport inference.

Moreover, the majority of the studies in Table 2.1- 2.2 have been conducted using supervised algorithms. Developing semi-supervised or unsupervised approaches enables researchers to take advantage of large amounts of unlabeled data in smartphone-based travel surveys. Hence, more research is required to examine semi-supervised and unsupervised approaches in mobile phone data analysis.

Regarding trip purpose, as mentioned in Table 2.2, there are fewer studies inferring it from mobile phone data. Based on studies reviewed in Table 2.2, the majority of research on trip purpose detection are conducted on small-size datasets.

Unlike mode of transport, trip purpose occurs at a specific geographic location, the trip destination, while the mode of transport can be characterised by all the GPS points along the trip trajectory. Due to the correlation between mode of transport and trip purpose, inferring trip purpose can be achieved by using both features, i.e. disaggregate features related to GPS trajectory and features related to destination location, such approach have been analyzed in Chapter 6.

One of the advantages of deep neural network models, such as RNNs or CNNs is that they can

handle both aforementioned feature types. That is, we can infer trip purpose analyzing trajectory related features, and socio-demographic or features related to the destination. Examining the performance of such deep neural network models is not addressed in the current literature and deserve researchers' attention.

In reviewing the literature on transit itinerary, we found just one study that analysed GPS trajectories to infer transit itinerary using rule-based methods. Zahabi et al. [82] implemented a rule-based algorithm on a GPS travel survey to detect transit itinerary. They used GTFS data to find a set of nearest active transit routes around each GPS point, and then inferred transit route by checking the history of all GPS points along a trip. While finding the demand for transit network is important information in transportation network design, there is a need for data collection efforts and artificial intelligence modeling experiments to infer transit itinerary from smartphone-based travel surveys.

Also, the literature lacks attempts at multi-task learning, despite the correlation between different trip characteristics such as mode and purpose. This is not to say that multi-task learning has not been used at all in transportation. Multi-task learning has been used in traffic data imputation studies [83]. Rodrigues et al. [83] proposed multi-output Gaussian processes (GPs) to model the spatial and temporal patterns in traffic data. They demonstrated that the multi-output GP model is able to capture the complex dependencies and correlations between nearby road segments to improve imputation accuracy. Their multi-output model outperforms other imputation methods, such as Recurrent Neural Network model, by taking into account the complex spatial dependency between subsequent road segments. Almost all studies reviewed at the time of writing this thesis, have tested the single-task mode or purpose inference. Hence, testing the multi-task learners to predict mode of transport and trip purpose from mobile phone data is another topic worth to focus on.

Based on the above summary of the literature, this study attempts to address the limitations in the literature in the following directions:

- Exercising the deep learning methods able to analyze the disaggregate point-based features to infer mode of transport and trip purpose
- Examining a semi-supervised method to infer mode of transport
- Using sequence labeling approaches, such as Recurrent Neural Networks, that can capture

the relation between sequences of GPS points along a trip trajectory.

- Predicting mode of transport and trip purpose via a multi-task learning.
- Inferring transit itinerary using a machine learning method.

Beyond the above methodological directions, there are also some other novel additions to the literature addressed in this thesis:

- Examining the application of location-based social media data, such as Foursquare and Google Place, to infer mode of transport.
- Testing the application of Google Transit Feed Specification (GTFS) data on mode of transport inference.

2.2 Background on Models Used in This Thesis

This section explains the background of the models used in this thesis. To do so we review the following machine and deep learning approaches:

- Random Forest model
- Convolutional Neural Networks
- Semi-supervised Generative Adversarial Networks
- Recurrent Neural Networks

Based on the categorization of artificial intelligence methods in Figure 1.4, the models developed in this study can be categorized as shown in Table 2.3. All the Random Forest models developed in Chapter 3 are supervised single-task trip-based learners. The CNN models in Chapter 4 can be categorized as supervised single-task point-based learners. The GANs models developed in Chapter 5 are semi-supervised single-task point-based classifiers. Finally the LSTM and GRU learners in Chapter 6 are supervised multi-task learners that are fed with both point-based and trip-based features.

Table 2.3: Categorization of the Models Developed in this thesis

Model	Categories (As explained in Section 1.3 and Figure 1.4)					
	Supervised	**Semi-Supervised**	**Single-task**	**Multi-task**	**Point-based features**	**Trip-based features**
Random Forest in Chapter 3	✓	-	✓	-	-	✓
Convolutional Neural Network in Chapter 4	✓	-	✓	-	✓	-
Generative Adversarial Networks in Chapter 5	-	✓	✓	-	✓	-
Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU) in Chapter 6	✓	-	✓	✓	✓	✓

We use Random Forest models to infer mode of transport, trip purpose and transit itinerary, since they have been shown to have good performance in many studies across many applications. The CNN model is able to analyse the disaggregate point-based features. Moreover, it has shown excellent results in image recognition studies during last years. Semi-supervised GAN models are also tested as a semi-supervised approach. The convolutional GANs model in this thesis can also be trained on disaggregate point-based features. Finally, we implemented Recurrent Neural Networks as they have the capability of being fed disaggregate point-based features, and moreover, they can consider the relations between the GPS points along a trip.

2.2.1 Tree-based Models

The Random Forest model was developed by Breiman [84]. The RF method is based on a combination of decision tree predictors. However, unlike the decision trees, it has no pruning or stopping rule [85]. The RF method first generates a series of training samples (T_K) from the original training dataset (dataset T in 2.1 with bootstrapping [84]. Then, about one-third of each bootstrap training sample is left out, named as the Out-of-bag (OOB) set, and a decision tree is constructed on the remaining observations (about two-thirds of the bootstrap sample, named as InBag set) [84].

The constructed decision trees are generated with "random split selection" [86] where a random selection of attributes is used at each node to determine the split [84]. Finally, these decision trees will later vote to form the bagged predictor [84]. As Breiman [84] has explained, the OOB sets are used to compute the prediction error rate of each tree. The average of these prediction error rates is reported as the generalization error rate of the RF model. There are two user-defined parameters

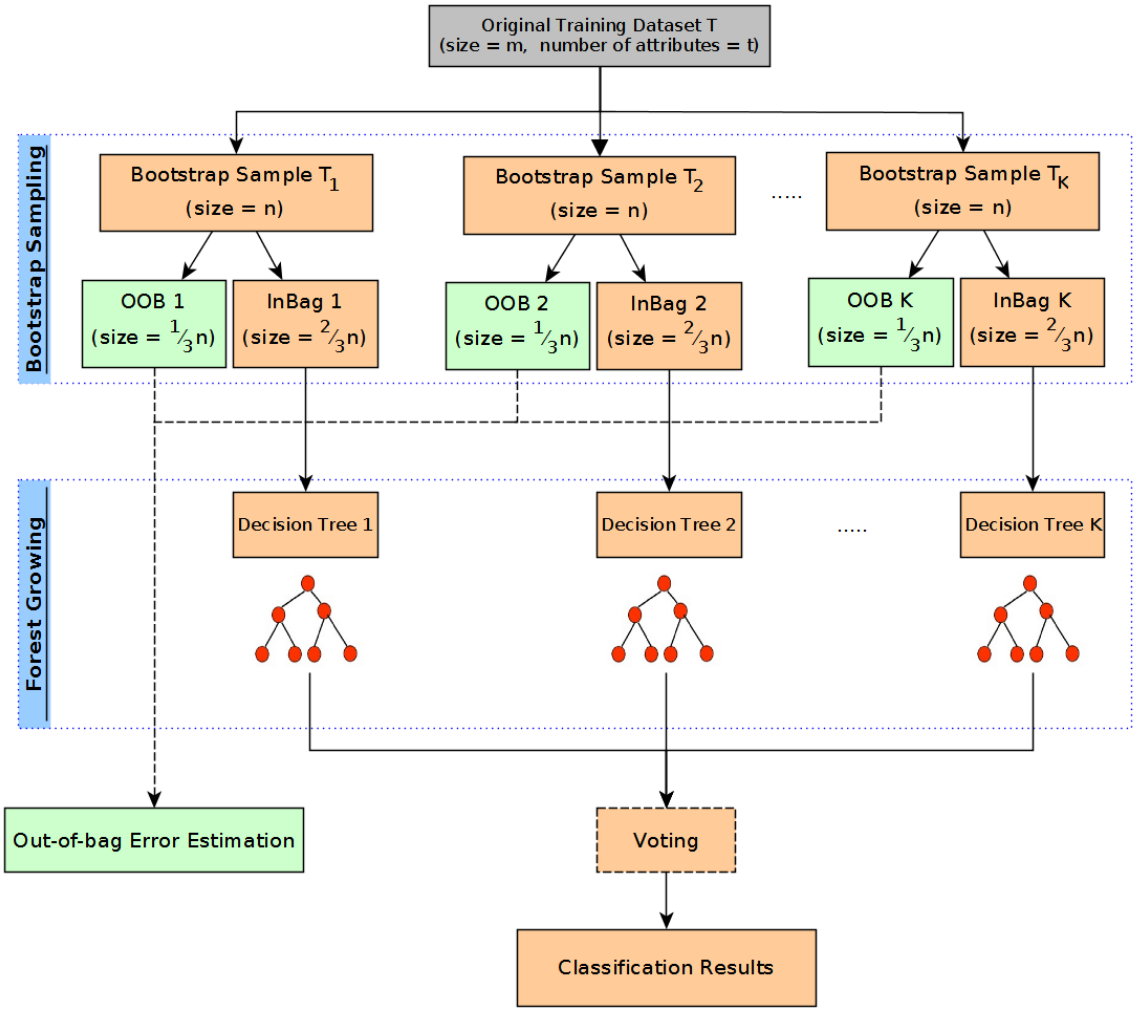


Figure 2.1: The flow chart of Random Forest for classification

in growing a forest. First, the number of randomly selected attributes used for each split. Second, the number of trees grown in the forest. Liaw and Wiener [87] suggested that for stable estimates of variable importance, a great number of trees is essential. The importance of each attribute in classification is useful information many researchers need to know. Several ways are suggested in the literature to measure the variable importance [85, 88]. The most well-known criteria are ”mean decrease in Gini“ and ”mean decrease in accuracy“ which have been used by many studies [87]. The mean decrease in Gini is calculated for the variable a_i (i is the total number of variables in the forest) as the average of all decreases in the Gini impurity where a split is formed by the variable a_i [85, 89]. Mean decrease in accuracy is calculated using the OOB observations as follows [85]: For each OOB_k set (k= 1,2 ...,K):

- Predict the class of each observation in OOB set using the decision tree constructed on the corresponding InBag set.
- Sum the number of times the decision tree classifies observations correctly.
- For each attribute a_i in the Random Forest:
    - Set the values of a_i equal zero in OOB set, call it the "permuted OOB set."
    - Predict the class of each observation in OOB set using the decision tree constructed on the corresponding InBag set.
    - Sum the number of times the decision tree classifies the observations correctly.
    - Subtract the number of votes for the correct class in "permuted OOB set" from the number of votes for the correct class in the OOB set.

Finally, the variable importance of a_i is calculated as the average difference in the accuracy of the OOB versus permuted OOB over the K decision trees [85]. We have applied the same RF model explained above to classify trip mode, purpose as well as transit itinerary.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks are a class of neural networks for processing grid-like topology data [33]. Examples of such data vary from 1D time-series data to 2D images. The underlying operation of Convolutional Neural Networks is affine transformation [90]. This involves a vector of inputs being multiplied by a matrix (also called a kernel or filter) to produce an output. In addition, a bias vector is typically added to the result of the matrix multiplication. Afterwards, the output is passed through a non-linear function, called an activation function. Typically, after the non-linear activation function, a pooling operation is applied (these operations are explained later). These stages, and how they are connected, are shown in Figure 2.2. The convolution operation is briefly described below. Also, different non-linear functions and pooling operation are described in Appendix C.1.

Generally, these mathematical operations form one "hidden layer" of a neural network. The output of each layer can be fed as input into the next layer. The final layer (also called "output" or

"classification layer") of a neural network in classification problems produces class probabilities by applying an activation function, such as the sigmoid or softmax functions. Figure 2.3 demonstrates the general architecture of CNN models.

The above mathematical procedures can be applied to any type of data, such as images, sound clips, or trajectories [90]. Regardless of the dimensionality, the input data for a neural network should possess three properties: (a) they can be stored as fixed-size multi-dimensional arrays (or tensors), (b) their characteristics can be presented along one or more axis, and (c) each axis (or variable), referred to as a channel, is used to store different information of the data. For example, an RGB (color) image usually has three channels: red, green and blue values for each pixel of the image [90]. In our case, GPS trajectories need to be converted to fixed-sized arrays with different channels.

2.2.3 The Convolution Operation

Convolution can be viewed as multiplication by a matrix. The dimensions of the matrix depend on the dimensions of the input data (also referred to as the input feature map). Figure 2.5 demonstrates an example of a convolution operation applied to a 1-D tensor, like the series of GPS points in our application. In this figure, a kernel steps by a "stride" of 1 across the 1-D tensor from left to right until it reaches the end [90]. At each step, the element of the kernel is multiplied by the input element it overlaps and the results are summed up to calculate the output at the current location of the kernel [90]. The boxes with arrows indicate how the left-most element of the output tensor is formed by applying the kernel to the corresponding left-most region of the input tensor. The convolution operation in Figure 2.5 is applied only to positions where the kernel lies entirely within the 1-D tensor. This kind of operation is called "valid" convolution, or non-zero padding[33]. The

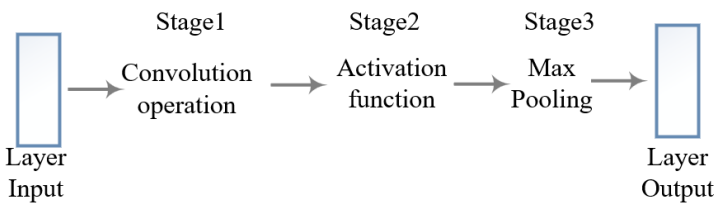


Figure 2.2: Typical Mathematical Procedures Involved in a Convolutional Layer

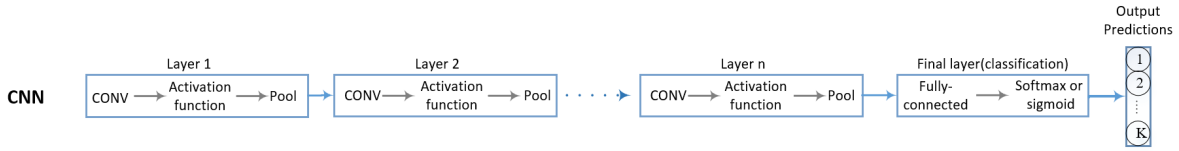


Figure 2.3: General architecture of a Convolutional Neural Network. (CONV: Convolution)

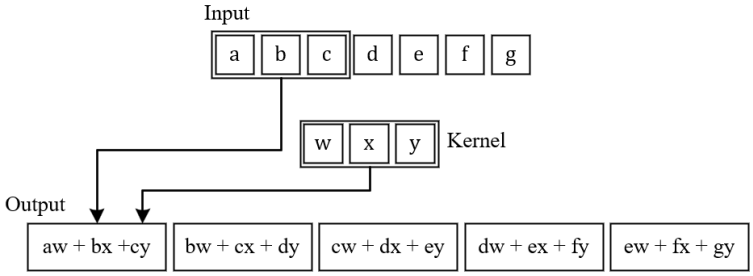
final output of this operation is called the output feature map [90]. This process can be repeated with different numbers and sizes of kernels. In Figure2.5, the kernel is defined by the $[w, x, y]$ tensor. The kernel weights, i.e. w, x, and y, are obtained through the minimization of the loss function. One thing to recognize is that without any other modifications to the input feature map, the dimensionality of the output feature map necessarily decreases with each convolution layer.

It is, however, often desirable for the dimensionality of the output feature map to be the same as the input feature map. To do this, "padding" is required. Padding involves concatenating zeros to the input feature map, and the beginning and end of a 1-D tensor. Figure 2.4b shows a convolution operation that shares the same properties of the convolution operation in Figure 2.4a, except that the padding is equal to 1. That is, a zero is added to the beginning and end of the tensor. This is referred to as "half padding" as opposed to "valid" padding described above.

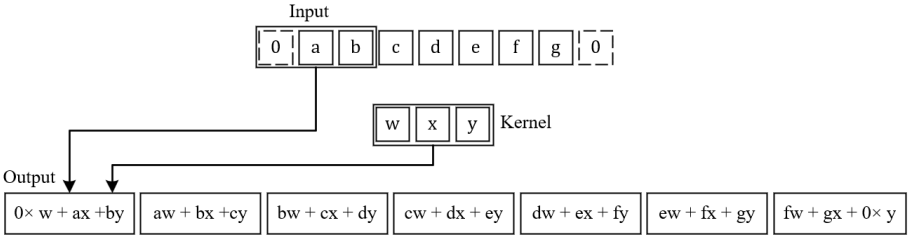
Four properties affect the convolution operation and the size of the output feature map (o) [90]: input size (i), kernel size (k), stride size (s) and padding size (p). For example, in Figure 2.5 the input is of size $[1 \times 7]$, and the kernel is a $[1 \times 3]$ tensor. Stride is the distance between two consecutive positions of the kernel [90]. The output size of a convolution operation can be obtained from the following formula [90]:

$$o = \frac{i + 2 \times p - k}{s} + 1 \tag{1}$$

Also, the number of channels for the output of a convolution layer is equal to the number of kernels used in that layer. There are two particular instances of zero padding that have been used widely in the literature because of their specific properties: half (same) padding and full padding. Half (same) padding is used when we require an output size that is the same as the input size. The padding size for half padding can be obtained from Equation 1 by setting $o = i$.



(a) An example of 1-D convolution with kernel size of 3, stride=1 and "valid" or "half" convolution over a GPS trajectory of size $[7 \times 1]$. The output shape is a tensor of size $[5 \times 1]$.



(b) An example of 1-D convolution with kernel size of 3, stride=1 and "half" padding over a GPS trajectory of size $[7 \times 1]$. The output shape is a tensor of size $[7 \times 1]$.

Figure 2.4: Convolution Operation on 1-D Input (Ex. A GPS Trajectory)

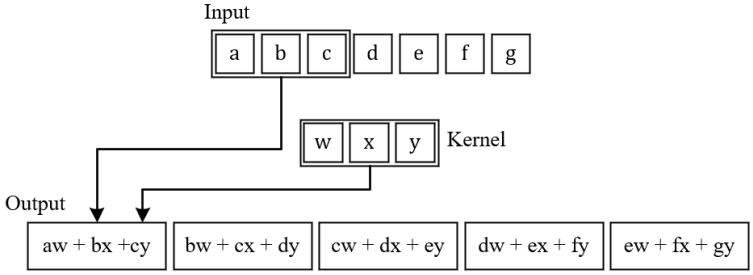


Figure 2.5: An example of 1-D convolution with kernel size of 3, stride=1 and "valid" convolution.

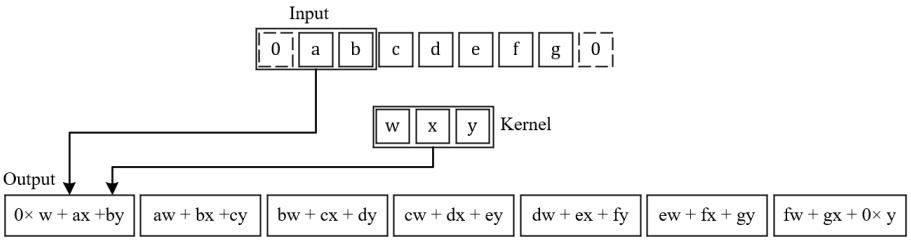


Figure 2.6: An example of 1-D convolution with kernel size of 3, stride=1 and half padding.

For example, Figure 2.4b shows half padding where kernel size is k=3, and the padding size has been set to 1.

2.2.4 Recurrent Neural Networks

Many learning methods make allowances for the dependency between observations. While this assumption, i.e. the independence of subsequent observations, has led to the rapid progress of learning models, it hinders taking into account long-range dependencies [91]. In many cases, artificial intelligence modeling aims to interact with a human being over time, such as AI-powered chatbots or recommendation systems, or virtual assistant software, like Apple's Siri.

It is not necessary for sequence data to be dependent over time. For example, in natural language processing (NLP), a sentence can be viewed as an ordinal sequence of words [91], without any explicit notation of time duration. Indeed, a great amount of literature in sequence labeling has been dedicated to natural language processing and speech recognition, where the dependency between data points is not necessarily temporal. Genomic data also can be regarded as DNA sequences. Predicting the properties and functions of DNA sequences is a valuable task in the broad field of genomics [92].

With respect to transportation applications, there are several cases in which data is inherently time-dependent. For example, trajectory data which consists of sequential GPS points is inherently time-dependent. Each GPS point along a trip trajectory has been recorded at a specific time.

The above-mentioned explanation describes an advantage of sequential labeling approaches, such as RNNs, over the approaches which mainly ignore the spatiotemporal dependencies, such as Random Forest, Support Vector Machines or logistic regression.

While sequences can be temporal, ordinal or spatiotemporal, a similar time-indexed notation can be denoted to all types of sequences [91]. A sequence of data points, ex. words of a sentence, activities of an activity pattern, frames of a video, can be presented as $(x^{(1)}, x^{(2)}, ..., x^{(T)})$, where $x(t)$ is a vector of attributes of each data point, and T is the length of the sequence where the sequence is finite. By applying a sequence model, each sequence of data points is mapped into a target sequence with a similar or different length [91].

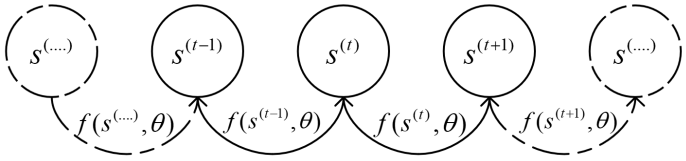


Figure 2.7: Unfolded Computation Graph of Equation 2

Terminology of Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks developed to analyze a sequence of data points. To take into account the dependencies between the data points in sequential data, the parameters in Recurrent Neural Networks are shared across different parts of the model. Researchers usually use computational graph language to explain neural networks [33, 93]. Each computational graph is accompanied by a set of allowable operations.

The difference between the computational graph of a neural network and a Recurrent Neural Network is that the latter includes cycles, which enable the graph to share parameters across deep network structures.

To include cycles in a computational graph, we first introduce the recurrent function. Equation 2 presents a recurrent function to define the status of a dynamical system at time *t*, $s^{(t)}$ based on its status at time *t*-1, i.e. $s^{(t-1)}$.

$$s^{(t)} = f(s^{(t-1)};\theta) \tag{2}$$

where, θ is the vector of weights for function *f*. Second, we unfold the Equation 2 by repeatedly applying the f function. The resulting expression can be represented in a directed acyclic computation graph.

The computation graph in Figure 2.7 does not contain the characteristics of input sequential data. Indeed, the state at time *t*, i.e. $s^{(t)}$, depends also on the corresponding input at time t, i.e. $x^{(t)}$ (as introduced in the previous section). In addition, at each time step *t*, the system may produce an output $o^{(t)}$.

To incorporate all the elements of a Recurrent Neural Network into a computational graph, the

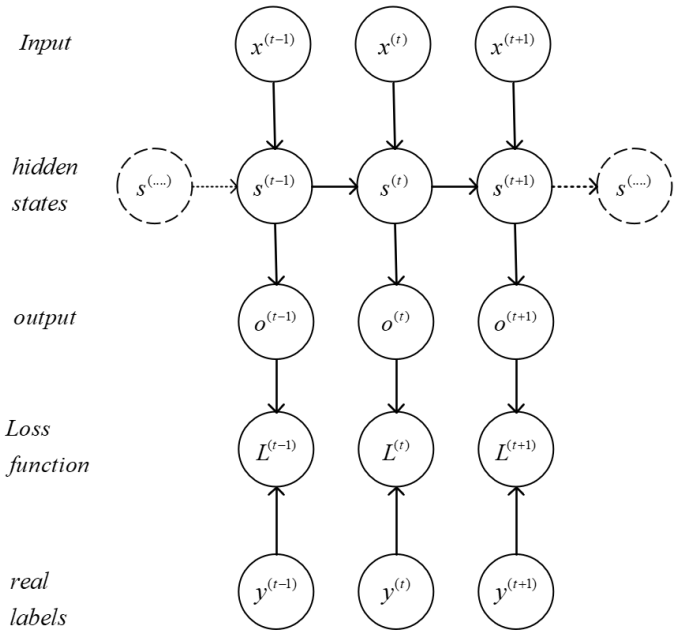


Figure 2.8: Computational Graph of a Recurrent Neural Network

following parts should be introduced. First, each data point $x^{(t)}$ in a training dataset has a corresponding target (or label) $y^{(t)}$. Second, the final goal of a Recurrent Neural Network is to generate outputs $o^{(t)}$ as close to the real targets $y^{(t)}$. To measure how far $y^{(t)}$ and $o^{(t)}$ are close to each other, researchers use a loss function $L^{(t)}$. Finally, the vector of weights to connect state at time *t*-1 to the state at time *t* is defined by θ and the vector of weights to connect each input $x^{(t)}$ to $s^{(t)}$ is defined by W. A full computational graph of a Recurrent Neural Network has been shown in Figure 2.8.

Challenges and limitations of conventional Recurrent Neural Networks

There are several challenges and limitations in conventional Recurrent Neural Networks that result in different approaches and architectures in developing Recurrent Neural Networks. We mention some of them here.

First, the conventional neural network only captures the impact of previous states on the current state of a system. For example, in a sequence of words fed into a conventional RNN, at each time step *t*, only the dependencies between the current word and the previous words are captured by the model. In other words, a link between hidden units of a conventional RNN has a delay of size one, as it begins from time step *t-1* to time step *t*. However, RNN can benefit from capturing

long-term dependencies, i.e. including links with delays d larger than one. Indeed, there are many cases, especially in speech recognition or language translation, that capturing the dependencies between all the neighbors in a sequence may improve the performance of sequence labeling. In such applications, conventional RNN which reads a sequence in an uni-directional way, from the beginning of a sequence to the current position, is not able to capture the dependencies between the current position and the next elements in the sequence.

Second, the conventional RNN is shallow in terms of transformations involved in any of its three main blocks, i.e. hidden-to-hidden, input-to-hidden and hidden-to-output. A shallow transformation is a transformation represented by only one layer of multilayer perceptron, i.e. an affine transformation followed by a nonlinearity function. Some studies have mentioned the inadequacy of depth in conventional RNN to perform the required mapping from input sequence to output [94].

Third, the conventional RNN is not able to map a sequence to a sequence with different length, they only maps a sequence to a sequence with the same length. There are many cases in language translation or other fields that the input and output sequence does not have the same length.

For any of the above-mentioned limitations and challenges, different approaches have been proposed in the literature. Furthermore, some studies have suggested different architectures that can enable us to overcome two or more of the above limitations. We have explained the most well-known RNN architectures in Appendix C.2.

2.2.5 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced by Goodfellow et al. [95] who formulated a GANs learning algorithm based on a game theory scenario. They trained generative models through an objective function that implemented a two-player zero-sum game between a discriminator D and a generator G [95]. The discriminator attempts to discriminate between real and fake input data, while the generator is optimized to generate input data (from noise) that "fools" the discriminator [95]. The "game" between the generator and the discriminator begins with the generator function that produces an example from random noise that is intended to fool the discriminator. As Goodfellow [95] explains, the generator can be conceived of as being like a counterfeiter, trying to make fake money. The discriminator acts like the police, trying to identify counterfeit money

from genuine money. To win in the game, the counterfeiter must learn to make fake money that is indistinguishable from genuine money, while the generator network must learn to create samples that are drawn from the same distribution as the training data [95].

GANs have been widely employed in image and text recognition, such as image generation [96], text to image synthesis [97], and image to image translation [98]. GANs were introduced primarily as unsupervised learners that benefit from setting up a supervised learning framework. However, besides unsupervised learning, GANs are also able to undertake semi-supervised learning. Salimans et al. [99] have added samples from the GANs generator G to their data set, as a new "generated" class, and then used a standard classifier to do the semi-supervised learning. So, if we consider a standard classifier for classifying a data point x into one of K possible classes, the semi-supervised classifier will classify the data point into K+1 classes, with the K+1th class containing the observations "generated" by a GAN [99].

Other studies have used GANs for multi-class labeling instead of having one "real" class and one "fake" class. Springenberg [100] has proposed Categorical Generative Adversarial Networks (CatGANs) for multi-class semi-supervised learning. In this framework, he considers a change in the GANs protocol. Instead of asking discriminator D to predict the probability of x being "fake," we can ask the discriminator to assign all examples to one of K classes, while assigning a "fake" label to output for samples from the generator G. With this change in the GANs protocol, the problem posed to the generator changes from "generate samples that belong to the real dataset" to "generate samples that belong to precisely one out of K classes" [100]. CatGANs was shown to exhibit classification performance competitive with state-of-the-art results for semi-supervised learning for image classification. Also, the study demonstrated the capability of the generator, which is learned alongside the classifier, to generate images of high visual fidelity [100].

Another semi-supervised GANs model to be developed is Conditional GANs. It endeavors to implement multi-class labeling, in a different way. As Mirza and Osindero [101] have explained, the unconditioned (traditional) generator in GANs has no control over the class labels of the data being generated. Mirza and Osindero, proposed a model able to direct the generating process by conditioning the generator on additional information [101]. They explain how GANs will be able to do

multi-class labeling by conditioning the generator on class labels. Their proposed model can generate samples for each class. For example, in image classification problems, you ask Conditional GANs for the horse class, and it draws you a picture of a horse.

As explained above, while GANs is mainly used as a generative model to generate samples through labeling the data into "fake" and "real" [102], some studies have extended GANs for multi-class labeling. However, almost all implementations of GANs models have been conducted in image processing or speech recognition studies. Analyzing GPS trajectories with semi-supervised GANs models to detect the mode of transport has not been applied in the literature, yet. In the next section, we describe the GANs framework and related concepts.

Concepts and Framework

A representation of the traditional (unsupervised) GANs framework is shown in Figure 2.9. The GANs training strategy is defined as a Nash equilibrium to a two-player, non-cooperative game [95, 99]. It consists of two "adversarial" models, each of which wants to minimize its own cost function, $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ for the discriminator and $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ for the generator. The Nash equilibrium is the condition where neither player can do better than the strategy of the other player. A Nash equilibrium is satisfied where $J^{(D)}$ is at its minimum with respect to $\theta(D)$ and $J^{(G)}$ is at its minimum with respect to $\theta(G)$. As Goodfellow [102] explains since each player's cost depends only on the other player's parameters, but each player does not have any control on the other player's parameters, the GANs framework is most straightforward to be considered as a game between two players, rather than as an optimization problem.

Both generator and discriminator models in GANs can be a mapping function that converts an input to an output through a non-linear transformation. In the original GANs framework (Goodfelow et al. [95]) the generative model captures the data distribution and the discriminator receives either a generated sample (from the generator) or a true data sample and must estimate the probability that a sample came from the training data rather than the generator. Indeed, we train the discriminator to maximize the probability of correctly labeling both training examples and samples from the generator. The training process of GANs consists of simultaneous gradient descent. On each step of the training, two minibatches are sampled. The first minibatch contains x values sampled from

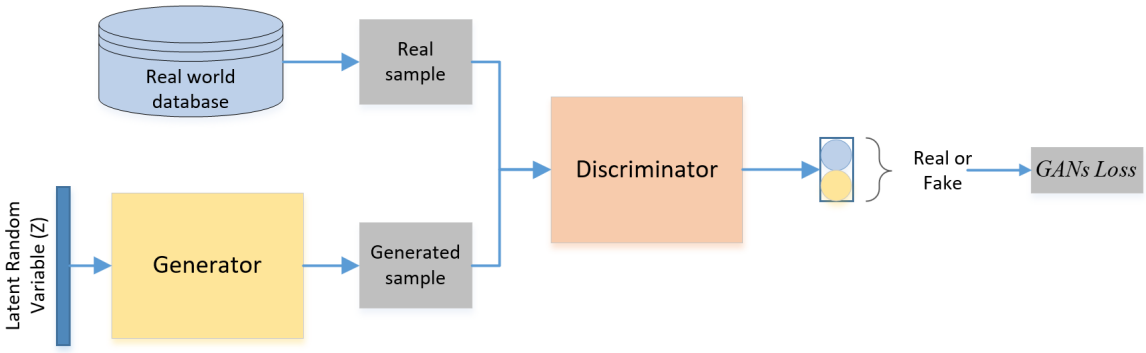


Figure 2.9: General Framework of Generative Adversarial Network.

the dataset, while the other minibatch consists of z values drawn from the model's prior over latent variables [102]. The gradient steps are applied on both minibatches simultaneously.

One updates the θ^D (the discriminator parameters), to reduce the cost function of the discriminator (J^D). The other updates θ^G (the generator parameters) to reduce the generator cost function (J^G) [102].

An important aspect of the GANs framework is the definition of cost functions used by the generator and the discriminator. In the original GANs framework, the discriminator is a binary classifier and usually uses standard cross-entropy cost. The cost function for binary classification is represented as the expectation (referred to as E in the following formula) taken across the training set, as [33]:

$$J(\theta) = -\mathbb{E}_{x,y\sim p_{data}} L[f(x,\theta); y] \tag{3}$$

where, L is the loss function, $f(x,\theta)$ is the classifier output when input is x, y is the true label of samples in the training dataset, and p_{data} is the empirical distribution, i.e. the distribution of training dataset. If we apply the cross-entropy cost function in Equation 3, we obtain:

$$J(\theta) = -\mathbb{E}_{x,y\sim p_{data}} ylog(f(x,\theta)) - \mathbb{E}_{x,y\sim p_{data}}(1-y)log(1-f(x,\theta)) \tag{4}$$

The only difference between the cost function of a binary classifier with the cost function of the discriminator in the GANs framework is that the latter is trained on two minibatches of data. One

minibatch comes from the training dataset where all the samples have label 1. The other minibatch contains the generated samples from the generator where all the samples have label 0. Hence, we can set $y = 1$ for the samples in the training dataset, and $y = 0$ for the generated (fake) samples from the generator. Also, the discriminator output for the samples from the training dataset is shown as D(x), and the discriminator output for the fake samples is represented by $D(G(z))$.

Finally, the discriminator cross-entropy cost function is explained with the following equation [95]:

$$J^D(\theta^D, \theta^G) = -\mathbb{E}_{x\sim p_{data}}[log\, D(x)] \; - \mathbb{E}_z\, log[1 - D(G(z))] \tag{5}$$

The generator cost function can be determined by the zero-sum game [95], in which the total cost of the discriminator and the generator should be zero. With this approach, the generator cost function can be defined as [99]:

$$J^{(G)} = -\, J^{(D)} \tag{6}$$

In the zero-sum game (sometimes referred to as the minimax game), the discriminator minimizes a cross-entropy while the generator tries to maximize the same cross-entropy [102], as in Equation 6. This is solved through gradient descent optimization.

In gradient descent optimization, the cost function eventually approaches zero when the classifier has chosen the correct class with high confidence. However, in the zero-sum game, when the discriminator correctly predicts the generated samples by the generator, the cross-entropy, i.e. $J^{(D)}$, approaches zero, and there is no chance for the generator to maximize the cross-entropy as its gradient already vanishes [95]. To solve this issue, Goodfellow [102] suggests "flipping" (multiplying by -1) the generator target used in Equation5, instead of flipping the discriminator cost function as in Equation 6. The author suggests using the following generator cost:

$$J^{(G)} = -\, \mathbb{E}_z\, logD(G(z)) \tag{7}$$

With the above cost function, the zero-sum game is no longer valid for the GANs framework (particularly $J^{(G)} + J^{(D)} != 0$). At the same time, it guarantees that each player has a strong gradient

when losing the game.

Semi-Supervised Generative Adversarial Networks

The GANs framework described in the previous section only differentiates between real and generated (or fake) samples and does not apply to multi-class labeling. In a neural network classification model, an input data point x is classified into one of K possible classes. Such a classifier is fed by x and outputs a K-dimensional vector of logits $[l_1, l_2, ..., l_K]$. Where logits refer to the raw predictions resulting from the last layer of a neural network, converted to class probabilities by applying the softmax function [99, 103]

$$P_{model}(y = j|x) = \frac{exp^{(l_j)}}{\sum_{k=1}^{K} exp^{(l_k)}} \tag{8}$$

Such a model is then trained by minimizing the cross-entropy between the true labels and the predicted class probabilities, i.e. $P_{model}(y = j|x)$.

Recently, Salimans et al. [99] used GANs for semi-supervised learning. The semi-supervised GANs framework (as shown in Figure 2.10) has a lot in common with the GANs framework for binary "real" and "fake" classification. The primary difference between them is the number of classes (k+1 instead only two) in the output. We can take advantage of semi-supervised learning with any standard classifier by simply including generated samples by GANs in our dataset, and assigning a new "generated" class label (y = K + 1) to them, while at the same time increasing the output dimension of our classifier from K to K + 1, as shown in Figure 2.10.

We can consider $p_{model}(y = K + 1|x)$ as the probability of x being fake. A semi-supervised model can also learn from unlabeled data, considering the fact that it corresponds to one of the K real classes by maximizing the logarithm of $p_{model}(y \in 1, ..., K|x)$ [99]. Considering that datasets in a semi-supervised context are made up of both generated and real data, the following loss function, proposed by Saliman et al. [99], is used to train the classifier:

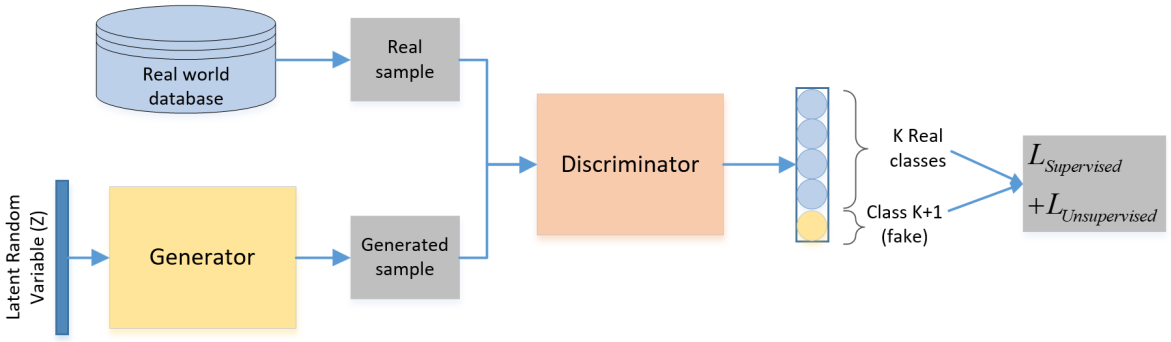


Figure 2.10: The Framework of Semi-supervised Generative Adversarial Network.

$$L = -\mathbb{E}_{x,y\sim p_{data}(x,y)}[log\ p_{model}(y|x)] - \mathbb{E}_{x\sim G}[log\ p_{model}(y = K+1|x)] \tag{9}$$
$$= L_{supervised} + L_{unsupervised}$$

where:

$$L_{supervised} = -\mathbb{E}_{x,y\sim p_{data}(x,y)}[log\ p_{model}(y|x, y < K+1)] \tag{10}$$

$$L_{unsupervised} = -\{\mathbb{E}_{x\sim p_{data}(x,y)} log[1 - p_{model}(y = K+1|x)] + \mathbb{E}_{x\sim G} log[p_{model}(y = K+1|x)]\} \tag{11}$$

In Equations 9-11, the total cross-entropy loss is decomposed into two components. The supervised loss $L_{supervised}$ is the logarithm of the probability of the labels of the real data (data from the training dataset). The unsupervised loss ($L_{unsupervised}$) is in fact the GANs discriminator loss function in Equation5. By replacing $D(x) = 1 - p_{model}(y = K+1|x)$ in Equation 11, we will have:

$$L_{unsupervised} = -\mathbb{E}_{x\sim p_{data}(x)} log\, D(x) - \mathbb{E}_z log(1 - D(G(z))) \tag{12}$$

This is the GANs discriminator loss function. The above cost functions in Equations 9-11 allows the GANs to learn a generative model and a classifier simultaneously. In this study we implement the same approach and present our empirical results to classify the mode of transport from GPS trajectories.

As Saliman et al. [99] have mentioned the optimal solution for minimizing the above loss functions, i.e. $L_{supervised}$ and $L_{unsupervised}$, is to have $exp[l_j(x)] = c(x)p(y = j, x)\forall j < K + 1$ and $exp[l_{K+1}(x)] = c(x)p_G(x)$, where $c(x)$ is an scaling function. By training G to minimize the GANs game-value, using the discriminator D defined by our classifier, we can train G to approximate the data distribution. Saliman et al. [99] have found that this approach introduces an interaction between G and classifier that can be used for semi-supervised learning. In this study we use the same approach. The mathematical procedures and architecture of GAN model are described in Appendix C.3.

Part II

Dissertation Articles

Chapter 3

An Automated Approach from GPS Traces to Complete Trip Information

Preamble

This chapter introduces a machine learning framework to predict the mode of transport, trip purpose and transit itinerary from MTL Trajet dataset. The goal of this chapter is to demonstrate the capability of a large-scale real-world smartphone-based travel survey to provide transportation planners with the most important trip information, as explained in Section 1.2, required in transportation demand analysis.

We used Random Forest model as a supervised learner to infer the most important trip information. As explained in Section 2.2, the Random Forest model is only able to process the aggregate trip-based features. After developing Random Forest models, the most important features in detecting each trip information are presented. We also compare our results against other modeling approaches in the literature. The Random Forest model developed in this chapter to infer transit itinerary is the first one in the kind that shows the capability of artificial intelligence to infer route-wise transit demand from trajectory data. This research article appeared in "International Journal of Transportation Science and Technology": Ali Yazdizadeh, Zachary Patterson, Bilal Farooq, "An automated approach from GPS traces to complete trip information", International Journal of Transportation Science and Technology, Volume 8, Issue 1, 2019, Pages 82-100, ISSN 2046-0430.

Abstract

Recent advances in communication technologies have enabled researchers to collect travel data based on ubiquitous and location-aware smartphones. These advances hold out the promise of allowing the automatic detection of the critical aspects (mode of transport, purpose, etc.) of people's trips. Until now, efforts have concentrated on one aspect of trips (e.g. mode) at a time. Such methods have typically been developed on small data sets, often with data collected by researchers themselves and not in large-scale real-world data collection initiatives. This research develops a machine learning-based framework to identify complete trip information based on smartphone location data as well as online data from GTFS (General Transit Feed Specification) and Foursquare data. The framework has the potential to be integrated with smartphone travel surveys to produce all trip characteristics traditionally collected through household travel surveys. We use data from a recent, large-scale smartphone travel survey in Montréal, Canada. The collected smartphone data, augmented with GTFS and Foursquare data are used to train and validate three random forest models to predict mode of transport, transit itinerary as well as trip purpose (activity). According to cross-validation analysis, the Random Forest models show prediction accuracies of 87%, 81% and 71% for mode, transit itinerary and activity, respectively. The results compare favorably with previous studies, especially when taking the large, real-world nature of the dataset into account. Furthermore, the cross validation results show that the machine learning-based framework is an effective and automated tool to support trip information extraction for large-scale smartphone travel surveys, which have the potential to be a reliable and efficient (in terms of cost and human resources) data collection technique.

3.1 Introduction

Travel demand forecasting and modeling are central tools in the analysis of transportation plans, projects, and policies in urban areas. During the last two decades, there have been rapid advances in the collection of data used in transportation modeling. In particular, the use of new technologies such as location-aware smartphones to collect data have become increasingly common [12, 13, 14]. These advances have enabled researchers to implement travel surveys with smartphone applications [14, 104] that gather respondent traces with lower costs, when compared with traditional travel surveys such as face-to-face interviews [6], Computer-Assisted Telephone Interviews (CATI) [105], or mail-back paper surveys [12]. Moreover, traditional surveys suffer from other shortcomings such as trip under-reporting [21, 22], time inaccuracies and origin-destination location errors, which generally caused by respondent fatigue or forgetfulness.

A great deal of research in this field has concentrated on methods of inferring trip characteristics (e.g. mode) from passively collected data. When this has been done, it often has considered one type of trip characteristic at a time, and has primarily used small or researcher-collected smartphone data; not on large-scale, real-world data collection efforts. The goal of this study is to show the potential of automatically detecting "complete" trip information based on data from a large-scale smartphone travel survey. We refer to "complete" trip information as including the following characteristics for a trip: geographic origin/destination, start and end time, mode, itinerary (route), as well as purpose the destination of a trip. These are the the characteristics typically required in trip-based transportation modeling [2].

In the literature, methods of origin/destination and mode detection have been studied extensively [4, 55, 69, 106]. By contrast, activity detection and itinerary inference have received less attention still [69, 82, 107], and the field is open to further research.

We believe that one reason for the lack of research on the two latter characteristics is their dependence on information apart from simple GPS traces of travelers. For example, transit itinerary detection depends on transit schedule and route information. Activity detection relies on comprehensive data from existing land-use as well as popular places around trip destinations [76, 79]. Recently, due to the widespread use of location-base social networks like Foursquare and Yelp [69, 79] a

comprehensive list of popular activities in urban areas is freely accessible and has the potential to be used in activity detection. Furthermore, General Transit Feed Specification (GTFS) data, which defines a common format to share public transportation schedules and associated geographic information [108], enables us to deploy detailed transit schedules and related locational information to infer transit itinerary from raw GPS data.

Apart from the data required, there is also the question of methodological approaches to infer information from smartphone and other data sources. Recently, there has been widespread use of machine learning classifiers, such as Neural Networks (NN) or Random Forests (RF), particularly in mode detection [55, 59]. Such success in mode detection encourages us to take advantage of such classifiers in transit itinerary and activity detection steps as well. In this study, we have developed a series of machine learning (Random Forest [84]) models (See Figure 3.1), to automatically detect mode, transit itinerary and activity from smartphone travel survey data. Also, regarding trip/segment detection we have used a rule-based algorithm developed by Patterson and Fitzsimmons [109].

The overall goal of this study is to show how it is possible to derive all critical trip information gathered by traditional travel survey methods such as CATI or mail-back surveys but through the processing of a large-scale smartphone travel survey data. This research has been conducted using data collected by the MTL Trajet [18], smartphone Travel Survey App. MTL Trajet was an instance of the smartphone travel survey app, DataMobile [109]. Datamobile was recently renamed Itinerum when it was built out into a travel survey platform in 2017 [8]. MTL Trajet was released as part of a large-scale pilot study on the 17th of October 2016 in a study that lasted 30 days. The location data collected from MTL Trajet is shown on a map of Greater Montreal in Figure 3.2. Also, a typical trip trajectory (sequence of GPS points) from MTL Trajet is shown in this figure.

The paper is organized as follows: a background section describes the literature related to GPS-based travel surveys and the algorithms used to process and derive information from them. The Methodology Section describes the methodology and algorithms used to detect the above mentioned "complete" trip information. The Data preparation section introduces the data collected in the MTL Trajet app during the study, as well as additional data used in the analysis. Afterwards, the model estimation section describes the estimated RF models and their attributes. The follwing section describes the prediction accuracy of the RF models based on the cross-validation results and

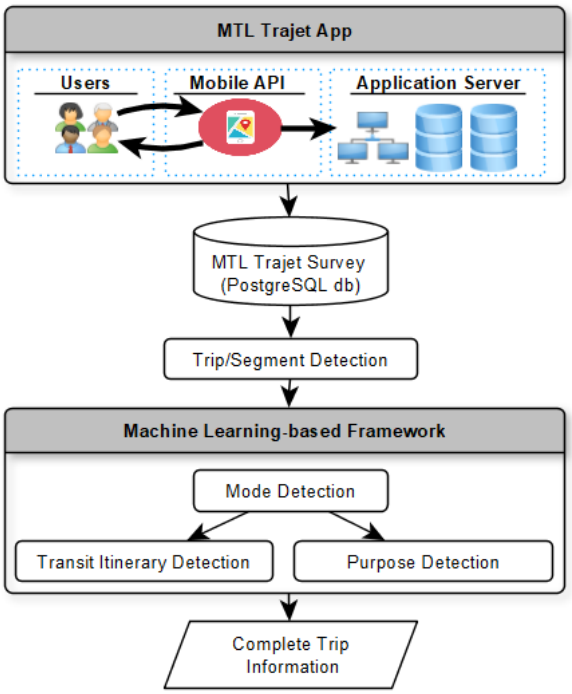


Figure 3.1: MTL Trajet App and Machine Learning-based Framework to Derive Complete Trip Information

compares these results with previous studies. Finally, we present our conclusions in the last section.

3.2 Related Background

Urban travel surveys were first introduced in the 1950s, where paper-based face-to-face interviews were conducted to elicit household trip information [3]. Disadvantages related to this method, such as labour and time costs caused them to be replaced by mail surveys in the 1960s [6]. The main drawback of mail surveys was their low response rates, which gave rise to Computer Assisted Telephone Interviews (CATI) as well as Computer-Assisted Self-Interviews (CASI) [46]. More recently, web surveys, which can be considered as a practice of CASI methods, have been used. However, both CATI and CASI suffer from non-response and misreporting [3].

Advances in Information and Communication Technologies (ICT) and the shortcomings of traditional travel survey methods brought about automated travel surveys based on GPS technology, beginning in the late 1990s [3]. In the beginning, GPS surveys were launched as a supplement to traditional surveys to increase their accuracy. However, the potential for GPS surveys in gathering

more accurate spatio-temportal characteristics of trips eventually caused them to be considered as a potential alternative to traditional surveys. Furthermore, by using prompted-recall surveys [110], which enabled travelers to validate their trips later on a website, GPS travel surveys grew in popularity. Nevertheless, even prompted-recall surveys face similar problems as traditional survey methods, such as the forgetfulness of travellers when validating their trips after the fact. Some studies, such as Xiao et al. [111], have implemented surveyor-intervened prompted recall surveys to enhance the accuracy of validations. In such surveys, the surveyors ask respondents by telephone to recall the details of their trips. However, this comes at the expense of rising the surveying cost. Recently, the trip validating step has been carried out in almost real-time by developing in-app prompts [8, 18, 112] that ask respondents to validate their trips as they go.

With respect to the theoretical concepts of machine learning algorithms, there is a considerable amount of literature in computer science as well as previously published transportation studies. Witten et al. [113] have clearly explained the machine learning algorithms such as decision trees, Random Forest, Support Vector Machines, etc., and their application in different fields of study. The book written by Goodfelow et al. [33] is a comprehensive reference to neural network algorithms and their applications. Regrading the transportation studies, Ghasri et al. [114] have explained in detail the decision tree and Random Forest classifiers. Dabiri and Heaslip [39] and Gonzales et al. [59] have described neural networks models and their application in transportation mode detection. In this section, we review studies using GPS-based travel surveys that also developed algorithm(s) for detecting trip mode, transit itinerary or activity. Table 3.1 and 3.2 report different approaches applied for trip/segment and mode detection, transit itinerary as well as activity detection, and their prediction accuracy.

3.2.1 Trip/segment Detection

With respect to detecting trip/segment, several studies have used rule-based algorithms, as shown in Table 3.1. Recently, Zhou et al.[115] have developed a Random Forest model to identify the trips ends using large-scale smartphone-based GPS tracking data. Their model achieved an accuracy of 96.17% for the identification of trip ends. With respect to the rule-based algorithms, dwell

Table 3.1: Studies on trip/segment and mode detection

Author/s	Method	Attributes	Validation Accuracy
Trip/Segment Detection			
Stopher et al.[46]	Rule-based	Dwell time	n/a
Wolf et al.[71]	Rule-based	Dwell time	n/a
Schuessler and Axhausen [49]	Rule-based	Dwell time, distance between points	n/a
Biljecki [48]	Rule-based	Dwell time, distance between points	91%
Gonzalez et al. [59]	Manually segmented by the cell phone user	Speed, acceleration, data quality, travel distance	n/a
Shafique and Hato [116]	Rule-based	Dwell time	n/a
Nitsche[55]	Rule-based	Speed, high amplitudes accelerometer signal	n/a
Dalumpines and Scott [50]	Rule-based	Dwell time and Speed	n/a
Xiao et al. [54]	Rule-based	Dwell time, Critical length, critical distance	96 %
Patterson and Fitzsimmons [109]	Rule-based	Dwell time, distance to metro stops, speed	n/a
Zhou et al.[115]	Random Forest	24 attributes including: instantaneous speed, acceleration, average absolute acceleration standard deviation of the instantaneous speed average heading change etc.	96.17 %
Mode detection			
Stopher et al.[46]	Rule-based	Speed, GIS, car/bike ownership	95 %
Bohte and Maat [47]	Rule-based	Average and maximum speed, GIS land use data	n/a
Schuessler and Axhausen [49]	Fuzzy-logic approach	Speed, Acceleration	n/a
Biljecki [48]	Fuzzy System	Proximity of trajectories to nearest net- work, speed	n/a
Gonzalez et al. [59]	Neural Network	Maximum speed, Average speed, Maximum acceleration, Average acceleration,Total Distance	91.23 %
Feng and Timmermans [43]	Bayesian Belief Networks	Speed, Accelerate, Car/ Bike/ Motor- cycle ownership, data quality	96 %
Shafique and Hato [117]	Random Forest	Acceleration and standard deviation, skewness, kurtosis, maximum and average acceleration	99.6 %
Nitsche et al. [55]	Ensemble of probabilistic and Discrete Hidden Markov Model (DHMM)	5th, 50th and 95th percentile of speed, accelerations, decelerations, direction change, standard deviation of the highfrequency accelerometer magnitudes, and power Spectrum of the accelerometer signal for Frequencies	Range from 65 % to 95 %
Dalumpines and Scott [50]	Multinomial Logit	Median speed, Median change in heading, total travel time	90 %
Xiao et al. [54]	Basyesian Network	Travel distance, average speed, average absolut acceleration, 95 % percentile speed, low speed rate, and average heading change	92.74 %
Eftekhari & Ghatee [44]	Rule-based	acceleration data from gyroscope and accelerometer sensors	95.2 %
Bantis & Haworth. [118]	Basyesian Network	Speed and socio-demographic data	Range from 82 % to 90 %
Endo et al. [58]	Deep Neural Networks	Time series of speed,head change, time interval, distance of the GPS points	Range from 84.8 %
Dabiri & Heaslip. [39]	Convolutional Neural Networks	Speed, acceleration, jerk (the rate of change in the acceleration) and bearing rate (rate of change in the heading direction)	Range from 84.8 %

time between GPS points has been considered as the most prominent criterion for detecting the segment/trips, ranging from 1 to 3 minutes. Additional rules have also been applied for trip/segment detection [3], such as point density [49], latitude and longitude change [105] or speed threshold and amplitude of the accelerometer signals [55].

Zhou et al.[115] have used several attributes as input for the Random Forest classifier, namely (1) local attributes such as time difference, instantaneous speed and acceleration, (2) global extreme

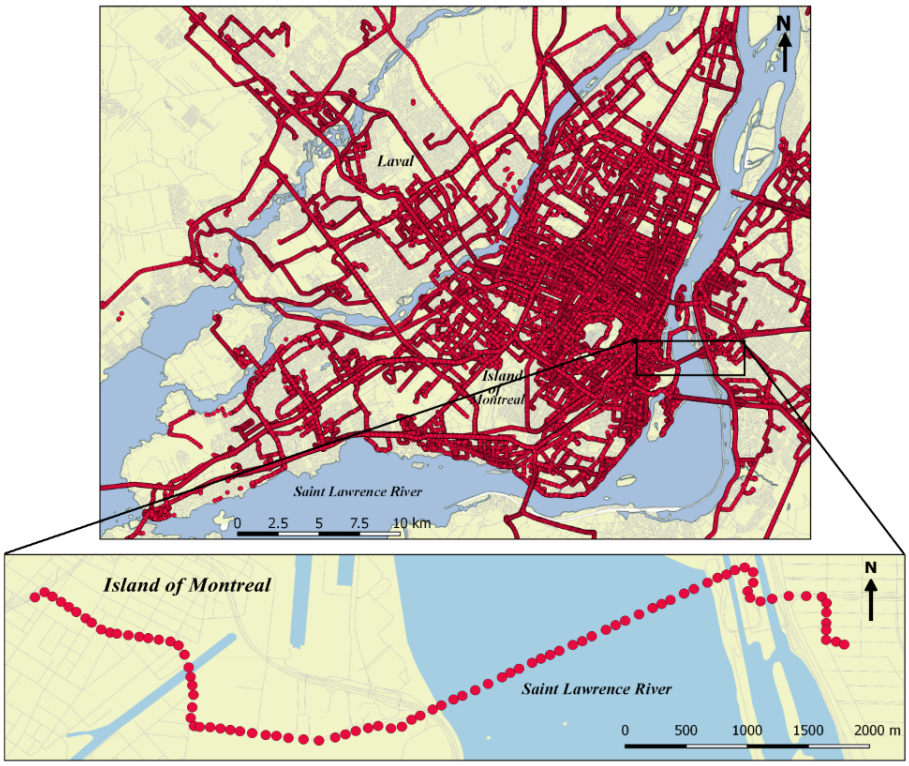


Figure 3.2: Location Data Collected in the MTL Trajet Study

value attributes like total duration and total duration, (3) speed-related attributes such as average speed and standard deviation of the instantaneous speed (4) acceleration-related attributes like the average absolute acceleration and the largest absolute acceleration, (5) tracking points clustering attributes, such as the largest distance between any two points within one neighborhood point set and (6) heading change attributes, like the average heading change in the neighborhood point set. They also proposed the concept of neighborhood point set to describe the temporally continuous points near a specific GPS tracking point.

3.2.2 Mode Detection

Mode detection, now well examined in the literature has been done using various approaches. Rule-based algorithms [46, 47], fuzzy systems [48, 49], machine learning classifiers [43, 54, 55, 59, 117] and discrete choice models [50] have been implemented. Mode detection algorithms have been applied on various data sources including raw GPS trajectories [37, 39, 64], smartphone's accelerometers [44], call detail records (CDRs) from smatrphones [119] , and smartphone's GSM data [120].

Moreover, some of the mode detection studies has exploited multiple data sources to improve the prediction accuracy of classifiers. Stenneth et al. [42] integrated the GPS and GIS information for developing a mode detection algorithm. Other studies [44, 55] have used gyroscope, rotation vector, and magnetometer data to improve the mode detection accuracy. In addition to the smartphone data, traveller's socio-demographics can result in higher prediction accuracy of classifiers [118]. A comprehensive and comparative review of existing studies on travel mode detection have been presented in the paper by Wu et al. [121].

Zheng et al. [64] developed a framework to automatically infer mode of transport from GPS trajectories. They have applied a rule-based segmentation algorithm to split a trip into segments with distinct modes of transport. Afterwards, various features such as mean and variance of the speed, top three speeds and accelerations have been utilized to develop classifiers, such as decision trees, to detect mode of transport. Sun and Ban [122] extracted acceleration- and deceleration-based features to classify vehicle type by Support Vector Machine (SVM) classifier.

Stenneth et al. [42] developed five classification algorithms (Bayesian Net, Decision Tree, Random Forest, Naïve Bayesian, and Multilayer Perceptron) to detect mode of transport. Their result demonstrated the superiority of Random Forest over other developed classifiers, in terms of prediction accuracy rate.

Recently, Xiao et al. [40] developed tree-based ensemble classification algorithms that outperform traditional ones such as the decision tree. Endo et al. [58] and Wang et al.0 [123] developed Deep Neural Network (DNN) algorithms to detect mode of transport. Also, Dabiri and Heaslip [39] have used Convolutional Neural Network (CNN) to train a mode detection model.

3.2.3 Activity Detection

Unlike mode detection, transit itinerary and activity detection have not received as much attention in the literature. With respect to activity detection, the methods in the existing literature have been categorized by Gong et al. [69] into three broad categories: rule-based algorithms, statistical methods and machine learning methods. However, recent studies in the field tend to use machine learning approaches along with geo-tagged information from social media, such as FourSquare or Google

Places [79]. Rashidi et al. [124] reviewed the studies on social media data and concluded that geo-tagged data possess a great potential to improve our knowledge in understanding activity behaviour. Geo-tagged social media data have been utilized by several studies from social science, computer science, and transportation science to extract meaningful activity behaviour patterns [124]. The studies include activity recognition [125], activity choice patterns [126], and understanding life-style behaviour from activity-location patterns [127]. Lee et al. [128] utilized geo-tagged tweets to generate individual activity spaces based on minimum bounding geometry (convex hull). Hasan and Ukkusuri [126] investigated Foursquare check-in data to infer individual weekly activity patterns using probabilistic topic models. However, as Rashidi et al. [124] have mentioned, the capacity of social media platforms such as Facebook, Twitter, LinkedIn, Foursquare, and Yelp to provide information on household daily travel has been minimally examined [124]. Zhang et al. [129] proposed a sequential model-based clustering method to investigate the potential of social media (Twitter) to realize the longitudinal household survey. They concluded that geo-tagged data (tweets) provide a sample of human activity space through a list of locations. Zhu et al. [78] investigated the potential of location-based social networks to explain the travellers' behaviour. They trained a Support Vector Machine (SVM) algorithm that achieved over 75% accuracy in predicting trip purposes combining with the traditional travel survey. Also, Lee et al. [128] reviewed studies on emerging data collection technologies for mode of transport and trip purpose prediction.

As demonstrated in Table 3.2, different attributes have been used in the literature to predict activity, such as information on land use types around each trip destination, related activity characteristics (e.g., duration of the activity at the destination, previous activity, activity start/end time), socio-demographics data (e.g., respondent's age, occupation and income), and Point of Interest (POI) data.

Oliveira et al. [76] developed a decision tree model to differentiate between 12 trip purposes. In their study, the decision tree model achieved a prediction accuracy of 65%. They also used a Nested Logit model that predict trip purposes with 60% accuracy. Wu et al. [4] developed two models, a rule-based and a Random Forest model to classify indoor, outdoor static (i.e., when an individual is relatively stationary while outdoors), out-door walking, and in-vehicle travel activities. Both of

Table 3.2: Studies on Trip Activity Detection and Transit Itinerary Inference

Author/s	Method	Attributes	Validation Accuracy
Activity Detection			
Stopher et al.[46]	Rule-based	GIS land-use data, home and workplace/school addresses, address of the two most frequently used grocery stores	over 60 %
Wolf et al. [71]	Rule-based	GIS land-use data	69 %
Bohte and Maat [47]	Rule-based	GIS land-use data, home and workplace/ school addresses	43 %
Wu et al. [4]	Random Forest	Distance, speed, acceleration	97% for indoor, 84% for in-vehicle travel, and lower for other
Lu and Liu [130]	Decision tree	Socio-demographics, land use, temporal information,previous & next trip attributes	73.4%
Pereira et al. [7]	Historical data matching rules	Activity duration, POI, socio-demographics, work hours travel time	n/a
Zhu et al. [78]	Support Vector Machine	Demographic Features: gender, age, occupation spatial Features: Foursquare venue category and tips temporal Features: duration stayed at a destination	75%
Oliveira et al.[76]	Decision tree	Land use, temporal information, socio-demographics	65%
Kim et al. [77]	Random Forest	Point of interest, age, gender	75.5%
Xiao et al. [111]	Neural Networks	Polygon-based information and point of interest	96.5 %
Zhang et al.[129]	Sequential model-based clustering method	Visiting frequency, most frequently -visited locations, distance between visited locations, relation between a location and its surrounding	n/a
Ermagun et al. [79]	Random Forest	Travel mode, previous activity type, trip characteristic, Nearby place characteristics, socio-demographics	64.17%
Transit Itinerary Inference			
Zahabi & Patterson [82]	Rule-based	Nearby bus routes (GTFS data), dwell time, speed	87 %

the models successfully predicted indoor activities and in-vehicle travel with 96% and 88% accuracy, respectively. However, the both models were fairly successful in identifying outdoor static and walking points. Kim et al. [77] developed a Random Forest model to differentiate between 16 purposes. However, they have distilled the set of 16 purposes into a set of 4 conceptually exclusive purposes: a) work (including work, education, and business), b) home, c) transportation-related activities (including pick up/drop off, and transfer or change the mode) and d) maintenance activities (including shopping, personal task, meal/eating break, health, etc.). They used age and gender variables in the modeling process to improve the prediction performance of the model. Their Random Forest model can predict the trip purpose by 75.5% prediction accuracy.

3.2.4 Transit Itinerary Detection

With respect to transit itinerary detection, we only found the study by Zahabi et al. [82] that implemented a rule-based algorithm on a GPS travel survey to detect transit itinerary. They used GTFS data to find a set of nearest active transit routes around each GPS point, and then inferred transit route by checking the history of all GPS points along a trip.

Based on this literature review, we note the following. First, studies typically consider only one trip characteristic at a time. Second, studies on mode and activity detection have been primarily based on small or researcher-collected smartphone data. For example, among 11 mode detection studies reviewed by Wu et al. [121] the sample sizes are less than 45 persons or less than 114 trips. Third, we observe that there is relatively little literature on detecting trip activity, and even less on transit itinerary. As such in this paper, we develop a framework to predict trip mode, activity as well as transit itinerary; that is, all trip characteristics typically required in trip-based modeling approaches, using data from the same large-scale study. In the activity detection step, we use FourSquare data as a standardized and highly accessible source for data on nearby places surrounding a trip destination. Also, in transit itinerary detection step, we use OpenTripPlanner router [131] to generate all possible transit options between each trip origin and destination.

3.3 Data Used

The following data sources were used to developing the models and algorithms to infer "complete" trip information from the MTL Trajet study:

(1) The MTL Trajet survey

(2) A Transit Itinerary survey

(3) Land-use Data

(4) Foursquare data

(5) General Transit Feed Specification (GTFS) data

(6) Bing Elevation data

A brief description of the above mentioned datasets is provide below.

3.3.1 MTL Trajet Survey

The purpose of the MTL Trajet study [18, 132] was to implement a large-scale pilot study by the City of Montreal to contribute to the development of their "Smart City" initiatives. The study was live from the 17th of October until the 17th of November 2016. In order to participate in the study, respondents needed to download the application, agree to the consent form, answer a few socio-demographic and daily travel-related questions and then allow the app to operate in the background of their phone. Respondents would be prompted after each stop to validate the mode and purpose of their trips. By the end of the study, there were 11,433 downloads of the Itinerum app, 4,780 on Android and 6,653 on iOS. Among them, 8,033 users responded to the socio-demographic and survey questions. In the end, there were 7,773 users for whom at least two data points were collected. These users provided data for 88,629 person days for an average of 11.4 days of participation per person. With respect to validations, 6,845 respondents validated at least one trip. Altogether, there were 131,777 validated trips, for an average of 19.2 validations per person.

3.3.2 Transit Itinerary Survey

The Transit Itinerary Inference (TII) data [82] was collected with the iOS version of the Datamobile app through a data collection initiative conducted at Concordia University in Montreal during July and August of 2016. The main goal was to collect validated transit itinerary data. There were 10 student surveyors who were asked to recreate transit trips identified randomly selected from the 2013 regional household survey in Montreal. The participants were asked to validate all public transit routes they used during the course of their trips. In the end, 599 validated transit segments were used in this analysis. We used this data set to train a Random Forest model to detect the transit itinerary.

3.3.3 Land-Use Data

Land-use information is one of the data sources with which the activities around a GPS coordinate can be inferred. The last updated Montreal land-use data was compiled by the provincial government ministry "MAMROT" [133] for 2011 for the "Montreal Metropolitan Community." This land-use data contains land-use characteristics of buildings. There are around 970 land-use types which have been classified into 23 categories. The 23 categories and the frequency of each land-use category has been shown in Table B.1 in Appendix B. However, as a large part of land-use parcels are residential buildings, using only land-use data may cause the activity detection algorithm to be prone to classification error, due to myriad residential parcels around trip destinations. Thus, we sought other location based data sources, such as Foursquare, to use them as a complement to land-use data.

3.3.4 Foursquare Data

Foursquare is an online location-based social network through which individuals can "connect" with the places they visit throuh "check-ins" using the Foursquare app. In general, a check-in specifies that a certain user has been present at a given venue. The check-ins are then associated with the venue as well as to other "friends" with the app [134]. For this study, for each trip destination, we sent a request to the Foursquare API to search all venues within 250 meter of a trip destination. According to the online Foursquare API documentation [81], each request to Foursquare API returns maximum of 50 venues. For each venue, there are 35 fields of information, among which the following were used in the current study, as they represent the most important activities/venues in the vicinity of a trip destination:

- Categories that indicate the Foursquare sub- or sub-sub-category to which the venue belongs
- Stats, which contains two pieces of useful information:

  (1) *checkinsCount*, which is the total check-ins ever in a venue

  (2) *usersCount*, which is the total users who have ever checked in a venue

As explained in the Foursquare API Documentation for Venue Categories [135] Foursquare has

categorized venues into the 10 top-level categories, as shown in Table B.2 in Appendix B. Each top-level category has "sub-" and "sub-sub-" categories that result in a total of 910 categories. In this study, we have aggregated all *checkinsCount* for each top-level category, giving 10 *checkinsCount* (as in Table B.2) for each trip destination. Also, the same procedure has been applied on *usersCounts* (as in Table B.2). This information is used in the Random Forest model to predict the trip purpose.

3.3.5 Bing Elevation Data

We used elevation data to approximate maximum and minimum slope of the earth along a trip. According to the documentation of Bing Elevation API [136], we can get elevations at equally-spaced locations along a path. By dividing the difference between the elevation of each two consecutive points by the direct distance between those points, we are able to approximate the slope of the earth between each pair of points. Then, the maximum and minimum slope along a trip were used in mode detection.

3.4 Research Design and Methodology

The framework developed to infer complete trip information, i.e. mode of trasport, trip purpose and transit itinerary, is illustrated in Figure 3.1. The MTL Trajet app was the data collection tool that gathered user location information and sent it to the MTL Trajet server. Afterwards, the data was transferred to a PostgreSQL database and cleaned through procedures explained below. Finally, the machine learning-based framework was developed to infer complete trip information. In this research, we have used one of the most common classifiers in machine learning, the Random Forest method, as it has shown to provide high prediction accuracy in many transportation classification situations [42, 79, 114, 116, 121]. Furthermore, since Random Forest is an ensemble learning approach, where predictions are made based on multiple decision trees, it is less prone to over fitting [79, 114]. The research includes four major steps: (1) data preparation, (2) model selection and attribute description, (3) model estimation, and (4) model accuracy validation. In this section, the first two steps are discussed. First, data preparation procedures are discussed. Afterwards, as the Random Forest (RF) model is the core machine learning algorithm in this study, a brief explanation

of Random Forest basics and related terminology is provided. Finally, the approaches used to derive complete trip information are explained.

3.4.1 Data Preparation

The MTL Trajet dataset contains over 33 million location (primarily GPS) points. To detect trips and segments we used the rule-based trip-breaking algorithm developed in Patterson & Fitzsimmons [109]. The algorithm detects segments based on 3-minute gaps in data while controlling for velocity and parameters relating to the public transit network (i.e. transit junctions and metro station location). Applying the trip-breaking algorithm on the MTL Trajet dataset resulted in 623,718 trips, of which 102,904 trips were validated by respondents.

Validated mode data was derived from the survey questions presented to respondents upon installation. In particular, respondents were asked the location of home, work and school, as well as the mode(s) of transport used for trips to these locations. Only trips from users who declared using only one mode option to travel between home and work or home and school were used. This procedure provided us with 10,518 validated trips. With respect to trip activity detection, six activity categories were used to predict trip purpose: "education," "health," "leisure," "shopping/errands," "return home" and "work." We used 102,904 prompt validated trips to train an RF model to detect the trip purpose.

3.4.2 Random Forest Model: Basics and Terminology

The RF method is based on a combination of decision tree predictors. The RF method first generates a series of training samples from the original training dataset with bootstrapping [84]. Then, about one-third of each bootstrap training sample is left out, called the Out-of-bag (OOB) set, and a decision tree is constructed on the remaining observations (the other two-thirds of the bootstrap sample is referred to as the InBag set) [84]. The newly constructed decision trees are generated with "random split selection" [86] where a random selection of attributes are used at each node to determine the split[84]. Finally, these decision trees will later "vote" to form the bagged predictor [84]. As RF model consists of multiple trees, the results are more stable than the individual decision tree models and less prone to overfitting [84].

The OOB observations are used to make a natural test set for calculating the error of each tree, rather than using the cross-validation method to estimate the error of the RF [85, 89]. In addition, the OOB sets can be used to calculate variable importance. The importance of each variable in classification is crucial information for the model builder to know. Several ways are suggested in the literature to measure the variable importance [85, 89]. The best-known criteria is the "mean decrease in accuracy", that has been used in several studies in the field [79, 85, 87]. The larger mean decrease of a variable, the more important that variable is deemed. The details of the calculation of the "mean decrease in accuracy" have been well explained by Breiman [89] and Archer [85]. For each tree in the Random Forest model, the importance of a variable V is defined as the decrease in the predictive accuracy on OOB data when the variable V is permuted. Overall variable importance is calculated as the average variable importance across all trees in the Random Forest. By definition, if variable V is not in a tree, its importance is set to zero.

Moreover, as Breiman and Culter [88] have explained, since the values of variable importance from tree to tree are independent, we can compute the standard error of each variable's importance. Finally, dividing a variable's importance by its standard error gives us a z-score that can be used to assign a significance level to each variable. Also, we can test the null hypothesis of zero importance for variable V and reject it where the calculated variable importance exceeds the -quantile of N(0,1). There are two hyperparameters in growing a forest. First, the number of randomly selected attributes used for each split. Second, the number of trees grown in the forest. Liaw and Wiener [87] suggested that for stable estimate of variable importance, a large number of trees is essential.

We have applied the RF model explained above to classify trip mode, activity as well as transit itinerary. All the RF models in current research were fitted using the randomForest package [87] in R version 3.4.0 [137].

3.4.3 Mode Detection

For this paper, a Random Forest (RF) model has been generated based on 10,518 validated trips from the MTL Trajet Survey. The mode of transport has been classified into five categories: walk, bike, car, public transit as well as car and public transit. Also, various attributes, summarized in Table 3.3, have been selected for generating the mode detection model. We have used trip characteristics such

Table 3.3: Attributes Used in Mode Detection Analyses.

Attribute	Definition
Trip characteristics	
CUM_DIST	Cumulative distance (meter) between O-D
DIR_DIST	Direct distance (meter) between O-D
TR_TIME	Travel Time (min.) between O-D
AVG_SPEED	Average speed (km/h) between O-D
85_SPEED	85th percentile speed (km/h^2) between O-D
MAX_ACC	Maximum Acceleration (km/h^2) between O-D
MIN_ACC	Minimum Acceleration (km/h^2) between O-D
DIFF_ACC	Difference between Min. and Max. Acceleration (km/h^2) between O-D
SLOPE_MIN	Minimum slope between O-D
SLOPE_MAX	Maximum slope between O-D
MAX_TIME_POINTS	Max time interval (min) between each consecutive pair of GPS point
MAX_DIST_POINTS	Max distance (meter) between each consecutive pair of GPS point
Time and Day characteristics	
TIME_DAY	Time of day from 0 to 24
DAY	1-7 for Monday to Sunday
Socio-demographics characteristics	
AGE	0: age between 16-24 ,1: 25-34 , 2: 35-44, 3: 45-54, 4: 55-65, 5: 65+
GENDER	0:male, 1: female, 2: other/neither
OCCUPATION	0: full-time worker, 1: part-time worker, 2: Student, 3: Student and worker, 4: Retired 5: At home
AVG_PRICE_NEIGH	The average municipality value of residential buildings around each individual's home (in 250 meters radius)
Geographical and closeness characteristics	
GTFS_ORIGIN	Direct distance between the origin and nearest public transit stop from GTFS data
GTFS_DESTIN	Direct distance between the destination and nearest public transit stop from GTFS data
CBD_ORIGIN	1: if the origin is located in Montreal's CBD, 0: otherwise
CBD_DESTIN	1: if the destination is located in Montreal's CBD, 0: otherwise

as 85th percentile speed, max/min acceleration and their difference, direct and cumulative distance. As traveler behavior varies on different days of the week and during different hours of a day, we also included time and day attributes related to a trip. Socio-demographic such as age, sex and occupation are the other attributes we considered in the RF model.

3.4.4 Transit Itinerary Inference

Automatic detection of transit itineraries from smartphone travel surveys is valuable to transportation planners for analyzing transit planning scenarios [82]. Our approach to infer transit itinerary is two-fold: first, it finds all possible transit itineraries between each origin and destination. Second, by training a machine learning model, the actual transit itinerary (i.e. the one chosen by the travelers) is detected. From a graph theoretical perspective, finding possible transit itineraries between an Origin-Destination requires a graph search algorithm, such as Dijkstra's [138], Bellman-Ford [139] or Multiobjective A^* [140, 141] algorithm. Such algorithms search among all possible paths for the

one that incurs the smallest cost (e.g. shortest travel time or minimum travel distance) [140, 141]. Detected itineraries are made up of a series of nodes and links from the transit and street networks. In this study, we have used the GTFS-based transit network, and the street network is taken from OpenStreetMap [142]. The graph search algorithm used is the one available through OpenTripPlanner (OTP) [131], an open-source, multimodal trip planning software system. OTP uses the Multiobjective A^* [140, 141] algorithm to find candidate transit itineraries (sequence of routes) between each origin and destination.

One approach to identifying transit itinerary among a set of possible alternatives is to match the trip trajectory and all candidate transit itineraries to find the itinerary chosen by the traveller. However, some problems arise with this approach. It is not always straightforward to identify the actual transit itinerary by matching the travel trajectories and all possible transit itineraries. First, GPS trajectories do not always perfectly match candidate transit itineraries due to errors in GPS recordings as well as signal loss in some urban areas or during underground trips. Second, when a transfer has occurred during a trip, there is a walking connection between the two transit routes which is not easy to detect, especially when transfers occur between bus and metro.

To solve such problems, some rules need to be applied. For example, a rule is required to specify the acceptable overlap (matching) percentage between the trip trajectory and alternative transit routes in order to say that the alternative route and the trip trajectory are the same. However, there is a wide spectrum of overlapping percentage values, from around 10% for metro trips where there is signal loss, or above 90% when the trip is done by bus without any transfer or signal loss.

Furthermore, we found that besides the overlapping percentage value, other attributes can help us to identify the chosen alternative, such as walking distance or waiting time during a transit trip. In addition, as the routing algorithm finds all possible transit routes between each Origin-Destination, sometimes it return unreasonable transit itineraries (e.g., far too long or circuitous) that can be easily rejected by setting a condition on the length or duration of transit itinerary. Hence, we decided to use a classifier, i.e. Random Forest, instead of a set of rules to identify transit itinerary. The attributes used in the RF model are shown in Table 3.4. The flow chart of the processing and estimation of the transit itinerary detection algorithm is shown in Figure 3.3.

Table 3.4: Attributes Used in Transit Itinerary Inference Analyses.

Attribute	Description
GPS tracks attributes	
GPS_AVG_SPEED	GPS tracks average speed
GPS_TIME	Time interval between the first and last GPS track of a trip
GPS_AVG_DIST	Average distance between consecutive GPS point
Attributes from OTP	
OTP_LEN	Itinerary length
OTP_TRANS_TIME	Total transit time of each returned (by OTP) itinerary
OTP_WALK_TIME	Total walking time of each returned (by OTP) itinerary
OTP_WAIT_TIME	Total waitingtime of each returned (by OTP) itinerary
OTP_TIME	Total travel time of each returned (by OTP) itinerary
OTP_TRANS	Number of transfers along each returned (by OTP) itinerary
OTP_WALK_DIST	Walking distance of each returned (by OTP) itinerary
OTP_ORDER	The order of itinerary returned by OTP
OTP_AVG_SPEED	Itinerary average speed
Attributes from GPS Tracks and OTP	
DIFF_LEN	Difference between GPS tracks length and itinerary length
OVL_PERC	Overlapping percentage of itinerary and GPS tracks

3.4.5 Activity Detection

The flow chart of activity detection algorithm is shown in Figure 3.4. The RF model is generated based on attributes shown in Table 3.5. The attributes come from three different data sources: MTL Trajet, land-use and Foursquare data. There are 61 attributes altogether, categorized into four major levels: (a) Socio-demographics, (b) Trip Characteristics, (c) Land-use, and (d) Foursquare.

3.5 Model Estimation

This section represents the results of mode detection, transit itinerary inference and activity detection models.

3.5.1 Mode Detection

As our basic/reference model, we started with the decision tree. However, as the prediction accuracy of decision tree models were around 70-80% and since the Random Forest model is an ensemble of decision trees and always achieves higher prediction accuracies, we continued our work on Random

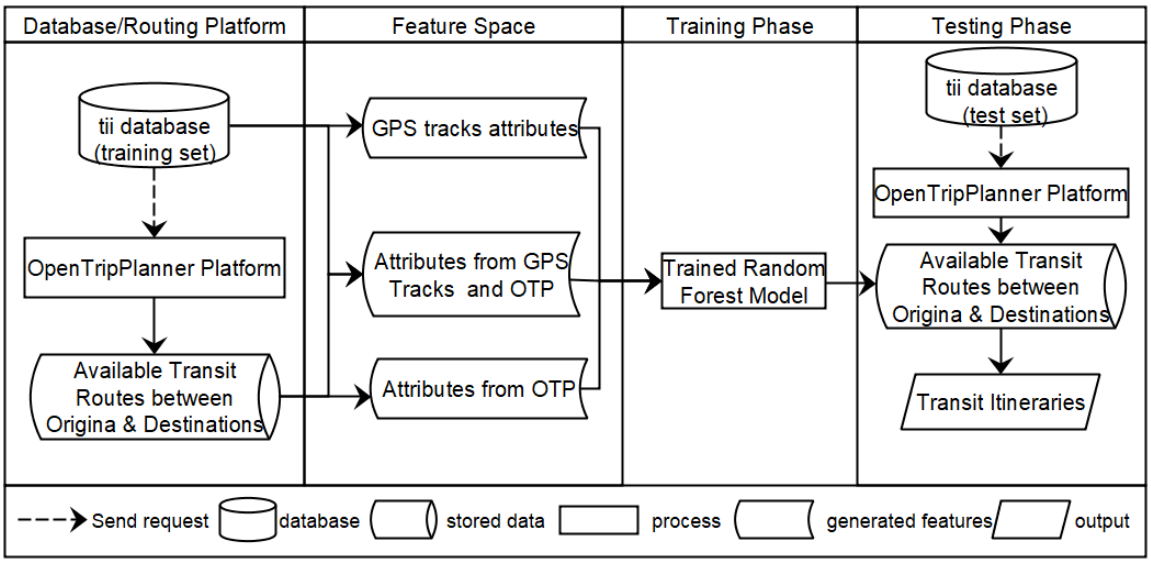


Figure 3.3: Flow Chart of Transit Itinerary Inference Model.

Forest models. To detect mode of transport, we generated Random Forest models with different numbers of trees, ranging from 100 to 2000. We observed no change in variable importance or prediction accuracy for Random Forests with more than 1000 trees. Also, we allowed up to 8 attributes to be randomly sampled at each split based on previously published recommendations [79, 84, 87]. Figure 3.5 demonstrates the variable importance plot pointing to how important each variable in classifying trip mode. The variable importance values have been calculated by dividing the raw variable importance by the standard error, as explained in Section 3.4.2. Hence, we use the these values to test the null hypothesis of zero importance for the variables at 95 percent confidence level (i.e. $|z| \geq 1.96$). Obviously, for all the variables in the Random Forest we can reject the null hypothesis, as all the values in Figure 3.5 exceed the z-score at 95 percent confidence value.

The most important variable, by far, is 85th percentile of speed. Average speed, direct distance, cumulative distance and "distance between trip origin and nearest transit stop" are the next most important variables. Among socio-demographics, average neighborhood price, as an indicator of income, is most important attribute.

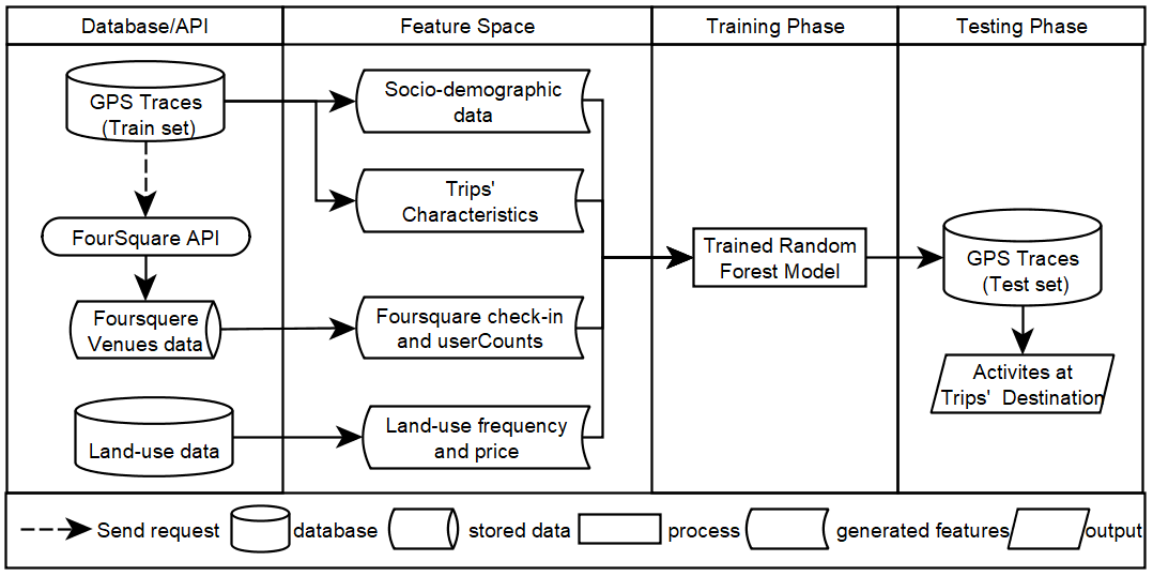


Figure 3.4: Flow Chart of Activity Detection Model

3.5.2 Transit Itinerary Detection

We developed one binary RF classifier to detect true transit itinerary among a set of transit itineraries generated by OTP. The RF classifier was allowed to grow with 1,000 trees, with 8 randomly sampled input attributes in each split. Figure 3.6 demonstrates how important (as explained in Section 3.4.2) each attribute is to labeling the itineraries. The most important variable is overlap percentage between actual chosen itinerary and the itinerary produced by routing algorithm (OTP). The next most important variables are waiting time, in-vehicle transit time, walking distance and number of transfers along each transit itinerary, pointing to the fact that travelers tend to minimize the waiting and in-vehicle time, walking distance as well as number of transfers. Also, we tested the null hypothesis of zero importance for the variables in Figure 3.6. As the importance of all the variables exceeds the z-score at 95 a percent confidence level, we can reject the null hypothesis.

3.5.3 Activity Detection

To detect the activities at trip destinations a RF model was developed using the attributes listed in Table 3.3. The RF model was generated with the same values as the two previous RF models for hyperparameters. Figure 3.7 shows the variable importance results of the RF model. Obviously, socio-demographic variables, i.e. age and average neighborhood price, as well as mode

Table 3.5: Attributes Used in Activity Detection Analyses.

Attribute	Definition
Socio-demographics	
AGE	0: age between 16-24 ,1: 25-34 , 2: 35-44, 3: 45-54, 4: 55-65, 5: 65+
AVG_PRICE_NEIGH	The average value of residential buildings around each individual's home
OCCUPATION	0:full-time worker, 1: part-time worker, 2: Student, 3: Student and worker, 4: Retired 5: At home
SEX	0:male, 1: female, 2: other/neither
Trip Characteristics	
DAY	1-7 for Monday through Sunday
CBD_ORIGIN	1: if the origin is located in Montreal's CBD, 0: otherwise
CBD_DESTIN	1: if the destination is located in Montreal's CBD, 0: otherwise
HOME_DEST	Direct distance between trip destination and individual home location
STUDY_DEST	Direct distance between trip destination and individual education location
WORK_DEST	Direct distance between trip destination and individual work location
HOME_ORG	Direct distance between trip origin and individual home location
STUDY_ORG	Direct distance between trip origin and individual education location
WORK_ORG	Direct distance between trip origin and individual education location
HOUR	Time of day from 0 to 24
MODE	Validated mode of transport via which the trip as been done
MTL_ORIGIN	1: if the origin is located in Montreal Island, 0: otherwise
MTL_DESTIN	1: if the destination is located in Montreal Island, 0: otherwise
TRAVEL_TIME	Total travel time of the trip
Land-use (number of land-use parcels in 250 meters around a trip destination)	
LU_*	23 different attributes each one shows the frequency of the corresponding land-use category
Foursquare (number of checkinCounts in 250 meters around a trip destination)	
CH_*	10 different attributes each one shows the checkinCounts for the corresponding Foursquare category
Foursquare (number of usersCounts in 250 meters around a trip destination)	
UC_*	10 different attributes each one shows the usersCounts for the corresponding Foursquare category

and "distance between destination and individual's home" are among the most important variables. Foursquare *checkinCounts* for "recreation," "transport infrastructure" as well as "art and entertainment" follow in importance. The next important variables are unoccupied land-uses and Foursquare *usersCounts* for "recreation". Also, the null hypothesis of zero importance for the variables in Figure 3.7 is rejected at the 95 percent confidence level.

In the next section, we have implemented a cross-validation procedure to validated the generated

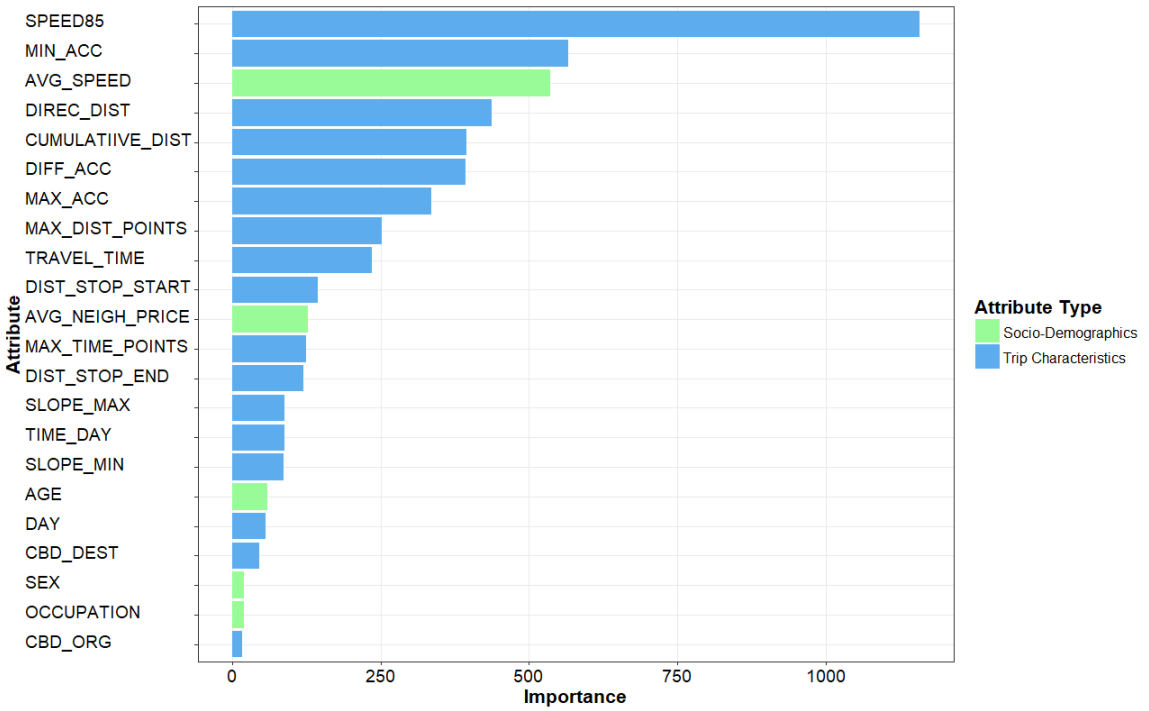


Figure 3.5: Variable Importance Plot Showing the Mean Decreases in Predictive Accuracy of Mode Detection Model.

RF models.

3.6 Discussion

This section presents the predictive accuracy of the different models as well as comparing the results with previous studies in the literature.

3.6.1 Prediction Accuracy Assessment

To assess the predictive performance of the RF models, k-fold cross-validation was used, where k = 10, as commonly adopted in the literature [59, 79]. The cumulative results of the 10-fold cross-validation for the RF models have been presented in a confusion matrix. Tables 3.6 to 3.8 show the analysis for the mode, transit itinerary and activity detection Random Forest models, respectively. The confusion matrices used in this section were generated based on the methodology described by Aggarwal [143]. Each column of a confusion matrix shows the total actual number of observations

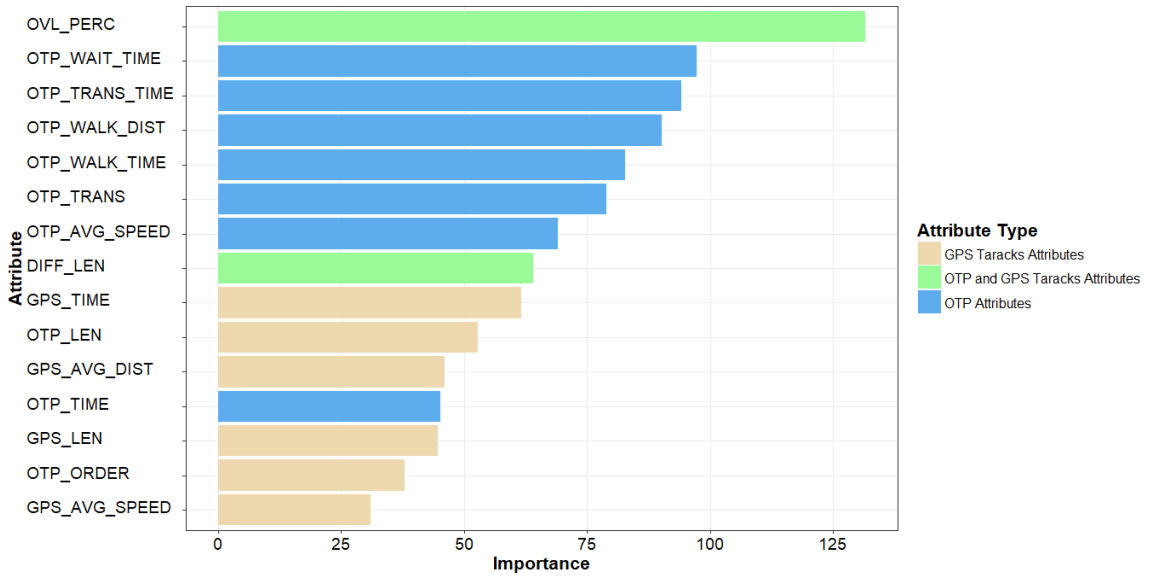


Figure 3.6: Variable Importance Plot Showing the Mean Decreases in Predictive Accuracy of Transit Itinerary Inference

for a given class over 10 iterations. In other words, each column indicates the total number of observations for a given class in the original dataset. On the other hand, each row is equal to the number of observations predicted by a prediction model to be of a given class [79]. Each confusion matrix also presents two complementary measures to assess the per-class accuracy of the predictive model: precision (positive predictive value) and recall (sensitivity) [143]. The precision value for a given class i indicates the percentage of observations predicted by the model as class i that are actually of class i in the test dataset. Also, recall that the value for a given class i represents the percentage of actual observations of class i that have been predicted by the model as class i.

For illustration, looking at the very first column of Table 3.6 indicates that of the actual 1,176 walk trips the mode detection RF model predicted 933 trips as walk, 41 trips as bike, 108 trips as public transit, 92 trips as car and 2 trips as "car and public transit". This indicates that 79.34% of the actual walk trips have been predicted correctly by the model. Similarly, looking at the very first row of Table 3.6 we can see that the mode detection model predicted 1,127 trips to be walk trips, out of which 933 were truly walk, 41 trips were bike, 95 trips were public transit, 53 trips were car and 5 trips were "car and public transit." This corresponds to a 82.79% precision performance of the mode detection model for walking. Overall, the prediction performance of "car," "public transit"

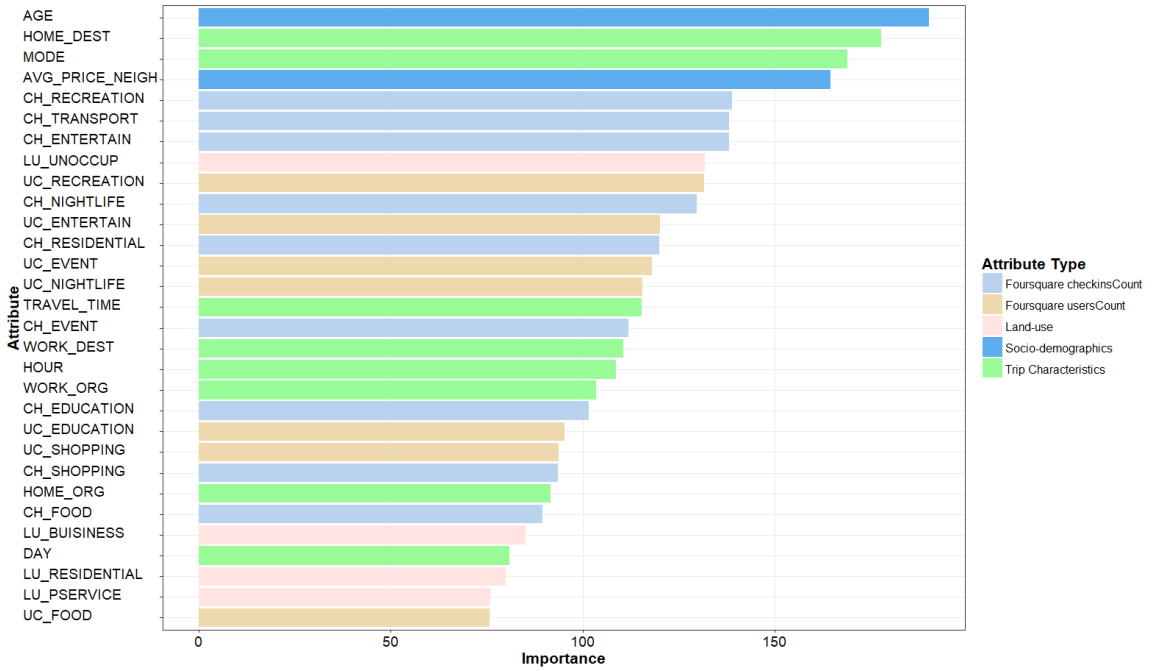


Figure 3.7: Variable Importance Plot Showing the Mean Decreases in Predictive Accuracy of Activity Detection Model (the first 30 important variables)

Table 3.6: Confusion Matrix Analysis for Mode Detection Model.

Trip Mode	Walk	Bike	Public transit	Car	Car and public transit	Precision(%)
Walk	933	41	95	53	5	82.79
Bike	41	527	31	48	6	80.70
Public transit	108	42	1907	146	83	83.42
Car	92	47	279	4660	37	91.10
Car and public transit	2	0	20	7	67	69.79
Recall (%)	79.34	80.21	81.78	94.83	33.84	

and "walk" trip modes is high both in terms of precision and recall. As we expected "Car and public transit" trip mode has the lowest precision and recall values, indicating this trip mode has similar attribute values to both "car" and "public transit." This caused these trips to be wrongly classified as car or public transit relatively often. Looking at the "Car and public transit" column, we see that of the 198 "Car and public transit" trips, 67 were correctly detected, While 83 trips were detected as car and 37 trips were detected as public transit, resulting in a low recall rate of 33.84%.

With respect to transit itinerary detection (Table 3.7), two classes, i.e. true and false itineraries, were validated. The third column of Table 3.7 shows the total number of transit trips in TII dataset.

Table 3.7: Confusion Matrix Analysis for Transit Itinerary Inference Model.

Class	False Itinerary	True Itinerary	Precision(%)
False Itinerary	566	111	83.60
True Itinerary	111	488	81.47
Recall(%)	83.60	81.47	

Table 3.8: Confusion Matrix Analysis for Activity Detection Model.

Activity	Education	Health	Leisure	Shopping/errands	Return home	Work	Precision (%)
Education	4,426	219	485	478	241	212	73.02
Health	61	1,902	74	47	42	50	87.41
Leisure	677	609	11,487	1,386	955	971	71.41
Shopping/errands	1,070	1,009	3,778	22,656	2,360	2,747	67.39
Return home	615	464	1,965	2,633	16,474	2,050	68.07
Work	328	513	1,020	1,656	862	16,382	78.91
Recall(%)	61.67	40.33	61.07	78.51	78.69	73.09	

Among the 599 transit trip segments in the TII dataset, 488 of them were validated correctly resulting in an 81.47% recall rate. Also, there were 677 not chosen, or false itineraries, among which 566 itineraries were correctly classified as False.

The confusion matrix for the activity detection model is shown in Table 3.8. The highest prediction accuracy belongs to the purpose health, with about 87% accuracy. However, looking at the "health" column in Table 3.8 indicates that among 4,716 trips with health as purpose in the MTL Trajet dataset, only 1,902 trips were correctly classified as "health" trips. This low recall rate tells us that a large number of health purpose trips are classified wrongly, predominantly as shopping trips, maybe due to the similarities between the attributes of these two activities or maybe because many health-related activities are accompanied by some shopping activities causing respondents to incorrectly validate the health purpose trips as shopping trips. The return home trip purpose has the highest recall rate in Table 3.8, meaning that 78.69% of return home trips in the dataset are correctly classified. The lowest prediction accuracy belongs to the shopping/errands activities, with 67% accuracy rate. This low prediction accuracy may be due to the similarity between shopping/errands trips and leisure trips, causing the RF model to predict shopping/errands as leisure trips. Also, the model has predicted a considerable number of return home trips as shopping/errand trips. This error may due to the fact that individuals usually do their daily shoppings on their way to home from work and at grocery stores close to home.

Summarizing the results, the mode, transit itinerary and activity detection RF models demonstrates

an overall accuracy of 87%, 81% and 71%, respectively.

3.6.2 Comparison with the previous studies

Comparing prediction accuracy rates across studies needs to be done carefully because of differences in sample sizes, number of classes, and the quality of data across studies [79]. As mentioned in Section 5.2, many studies in the literature have been based on small or researcher-collected smartphone data. In addition, the validation process can also effect the prediction accuracy of the models. Validation processes in smartphone travel surveys are usually done through an in-app prompt, a prompted-recall survey or a combination of them. Validating trip characteristics through a prompted-recall surveys can enhance the accuracy of reported trip characteristics and has a critical effect on the prediction accuracy of classifiers. The recall is usually implemented on the internet, when respondents are required to correct their trip characteristics if necessary [111]. Surveyor-intervened prompted recall surveys have also been used in some studies [111]. In such surveys, the surveyors ask respondents by telephone to recall the details of their trips.

Also, the number of categories varies across studies, ranging from very coarse (e.g. motorized/stationary [44]) to relatively fine (e.g. walk/bike/bus/metro/car [39]). Such differences are also observed across activity detection studies, ranging from very general (e.g. indoor/outdoor [4]) to fine (e.g. work/education/shopping/eating out/recreation/Personal business/return home [79]). Such differences need to be considered when comparing the prediction accuracy of classifiers.

Furthermore, smartphone travel surveys varies regarding data from mobile phone sensors, such as GPS, accelerometer, gyroscope, rotation vector and magnetometer [44, 45] all of which have been used when detecting mode of transport [39, 43]. However, the larger the number of sensors that are used by an app, the greater the impact on battery life on respondent devices [54]. In addition, middle to low end smart-phones are not usually equipped with the all sensors. Hence, for large-scale applications, using few sensors is often optimal [39, 54, 118].

Taking these considerations into account, our results compare well with other research in the literature. With respect to mode detection (Table 3.6), we have chosen studies that have used data from GPS sensors and whose validation process is similar to the in-app prompt validation process of the MTL Trajet data set. Dabiri and Heaslip [39] have reported test accuracies of 84.8% and 78.1%,

on the GeoLife dataset [144], for their Convolutional Neural Network and Random Forest models, respectively. Their best model, is able to predict walk, bike, bus, driving and train, with 81.6%, 90.3%, 80.7%, 86.6%, 92.3% precision, respectively. Also, the study of Zheng et al. [64], which is the first solid mode detection framework from 2008, reported accuracies of 89%, 86%, 66% and 65% for the walk, driving, bus and bike modes, respectively. Their decision tree-based inference model, results in an overall accuracy of 76.2%. Both the aforementioned studies [39, 64] have used GPS data from mobile phones, similar to the MTL Trajet. Moroever, they have small sample sizes of 69 and 65 persons, respectively.

The Random Forest model in this study shows an overall accuracy of 87% and can predict walk, bike, public transit, car as well as car and public transit with 82.79%, 80.70%, 83.42%, 91.10% and 69.79% precision, respectively. The results demonstrates that overall accuracy of our Random Forest model is higher than the overall accuracy of both aforementioned comparable studies, although regarding bike trips our accuracy is lower than the corresponding accuracy in Dabiri and Heaslip's study and higher than that of Zheng et al [64].

Also, Gonzalez et al. [69] produce similar accuracy rates of 92% for car trips and 81% for bus trips, where our mode detection RF model predicts car trips with 90% and public transit trips with 83% accuracy rate.

Bantis and James Haworth [118] have developed several models, including a Random Forest model, to detect three modes of transport (walk, bus/car and train) as well as stationary points from GPS and accelerometer data as well as users characteristics. Their training data contains trips from 5 individuals. They have proposed a hierarchical dynamic Bayesian network and compared the results against Random Forest, SVM, and Multilayer Perceptron classifiers. The accuracy of their proposed model is 90%, while it can predict Stationary status as well as Walk, Bus/Car and Rail trips with 94%, 65%, 88%, 91% precision, respectively. Although their categories are different from ours and the sample size of two studies are not similar, our Random Forest model predicted walking trips with higher precision, i.e. 65% vs. 82.79%. However, the overall prediction accuracy of our model is about 4% lower than their result, maybe due to lack of accelerometer data in our data set.

Eftekhari and Ghatee [44] have applied different models to detect the movement and the stationary statuses in the motorized and non-motorized modes on a smartphone travel survey of 9 users. Their

data set contains data from accelerometer, magnetometer and gyroscope sensors. Their best inference model can recognize the motorized mode with 95.2% accuracy. However, their result is not comparable with the results of this study or other studies with dissimilar categories.

We could not find any result of accuracy for "car and public transit" trips in the literature. Our Random Forest model can predict this mode with 69.79% precision, which shows that more work is needed regarding detecting multimodal trips.

With respect to activity detection (Table 3.8), the Random Forest model developed by Ermagun et al. [79] produced an overall accuracy of 64% for predicting 5 trip purposes.They have reported the precision of 50.10%, 61.49%, 51.92%, 55.62% and 47.04% for predicting Eat out, Education, Personal business, Shopping, as well as Social, recreation, and community, respectively. Our Random Forest model is able to predict Education, Shopping/errand and Leisure with 73.02%, 67.39% and 71.41% precision, respectively and shows an overall accuracy of 71%. Also, the RF model generated by Oliveira et al. [76] predicts activities with a 65% accuracy rate, which is lower than the result in this study.

Xiao et al. [111] have applied artificial neural networks combined with particle swarm optimization on a surveyor-intervened prompted recall survey to detect 5 trip purposes (Home, Work/education, Eating out, Shopping, Social visit, Picking up/dropping off someone). Their sample size is 352 respondents and their proposed model shows the accuracy of 96.5%, which is higher from our model. We think that, besides the differences between modeling approaches, the high quality of their validated data, gathered by a surveyor-intervened prompted recall survey, contributes to the high prediction accuracy of their classifier.

Regarding transit itinerary detection (Table 3.7), we found only one study in the literature, that of Zahabi et al. [82], who reported an accuracy of 87% of for their rule-based transit itinerary detection algorithm. The study reported correctly predicted distance of transit trips while our Random Forest model uses trip segment as unit of analysis. As such, the results of the two studies are not perfectly comparable.

3.7 Conclusion

This research contributes to the literature on GPS-based travel diary surveys in four important ways: (1) it demonstrates the potential of deriving complete trip information (i.e. not only commonly inferred trip characteristics such as mode, but also purpose and transit itinerary) from smartphone travel surveys; (2) it shows that socio-demographic characteristics of travellers can play an important role in predicting the mode and activity from smartphone travel surveys; (3) it also contributes to the literature of transit itinerary inference by introducing a new approach to infer transit itinerary from GPS data; and finally (4) it shows the advantages of using other complementary data sources such as location-based social network APIs, like Foursquare, or GTFS data alongside GPS traces to develop more accurate predicting models.

Although there are still some hurdles to overcome, we believe that smartphone travel surveys have the potential to replace traditional travel surveys. Here, we try to address some of these hurdles. First, large-scale smartphone travel surveys may not produce the same quality of validated data as small or researcher-collected smartphone travel surveys. Prompted-recall surveys can improve the quality of gathered data by reducing the self-report errors, however, this comes at the expense of rising the surveying cost and more burden on the respondents. Actually, the need for labelled data is a requirement of supervised machine learning models used in the literature and the current study. Using semi-supervised or unsupervised models may help to reduce or eliminate the need for labelled data. Although there are some studies [37] investigating semi-supervised approaches to detect mode of transport, more research is needed to adequately evaluate these methods in this context. Second, detecting multimodal trips is usually harder than detecting uni-mode trips from GPS traces. Future studies should also investigate multimodal trip detection methods. Third, the data from traditional household travel surveys are also have used in activity-based models [24] which need the information about tours of travellers as well as the sequence of their activities along a day. While a lot of studies in the literature aim to detect trip characteristics and single activities, there are few studies, like Lin et al. [119], aim to infer activity sequences as well as tours from smartphone travel surveys. Hence, more research is needed in the field to make the data from smartphone travel surveys applicable for activity-based modeling approaches.

The benefits related to smartphone travel surveys, such as lower costs and less user burden, provide encouragement for future research in the area to improve the prediction performance of models as well as overcome the above mentioned hurdles along the way. The results of current research demonstrate the potential of smartphone travel survey apps to be used as a stand-alone large-scale household travel survey in the future. Also, this paper shows that the trip characteristics extracted from GPS traces have the potential to be as complete as those of traditional survey methods, such as CATI or paper-based household travel surveys.

3.8 Acknowledgments

This research is based upon work supported by the Social Sciences and Humanities Research Council of Canada as well as “MonRéso Mobilité” research contract with Ville de Montreal.

Chapter 4

Ensemble Convolutional Neural Networks for Mode Inference in Smartphone Travel Survey

Preamble

This chapter provides an application of deep learning for mode of transport detection using disaggregate point-based features. In the previous chapter, Random Forest model was developed, to detect mode of transport, as a machine learning approach. However, as Random Forest model was developed using aggregated segment-based features, in the current chapter we deployed a deep learning model, i.e. Convolutional Neural Network(CNN), which is fed with disaggregated point-based features and enables us to analyze the detailed information of each GPS point along a trajectory. Moreover, due to the success of CNN models in other fields of study, particularly in image recognition tasks, there was a question about the application of such deep learning methods on the trajectory data. Hence, the other goal of this chapter is to show the potential of one of the well-known deep learning approaches in trip information inference. The CNN approach has been widely used in image recognition tasks to analyse the pixel-wise information of the images in 2-dimension space. However, the CNN models developed in the current study can analyze the point-based features in

trajectory data in a 1-dimensional space of GPS trajectories.

The models developed in this study are fed a smaller number of features compared to the Random Forest model in Chapter 3. Moreover, as we had focused on the point-based features, the socio-demographics have not used in the modeling procedure. Besides developing CNN models, we also investigated different ensemble methods on top of several Convolutional Neural Networks. We compared the results against other studies in the literature.

This research article appeared in "IEEE Transactions on Intelligent Transportation Systems":

A. Yazdizadeh, Z. Patterson, and B. Farooq. Ensemble Convolutional Neural Networks for Mode Inference in Smartphone Travel Survey. IEEE Transactions on Intelligent Transportation Systems, pages 1–8, 2019.

Abstract

We develop ensemble Convolutional Neural Networks (CNNs) to classify the transportation mode of trip data collected as part of a large-scale smartphone travel survey in Montreal, Canada. Our proposed ensemble library is composed of a series of CNN models with different hyper-parameter values and CNN architectures. In our final model, we combine the output of CNN models using "average voting", "majority voting" and "optimal weights" methods. Furthermore, we exploit the ensemble library by deploying a Random Forest model as a meta-learner. The ensemble method with Random Forest as meta-learner shows an accuracy of 91.8% which surpasses the other three ensemble combination methods, as well as other comparable models reported in the literature. The "majority voting" and "optimal weights" combination methods result in prediction accuracy rates around 89%, while "average voting" is able to achieve an accuracy of only 85%.

4.1 Introduction

In the last two decades, advances in data collection techniques, particularly in the use of new technologies such as Global Positioning Systems (GPS) and smartphones have been changing the options available for collecting transportation demand data. Using the data collected from these emerging technologies typically requires methods of data processing and inference. Detecting transport mode has received the lion's share of attention in the literature as researchers have tried to infer mode with methods ranging from rule-based approaches to artificial intelligence algorithms.

In recent years, deep neural networks have achieved considerable success in various applications, particularly in image recognition tasks. The performance of traditional machine learning methods are highly dependent on the choice of data representation (or features). Also a major part of efforts in applying them is spent on pre-processing data and extracting "hand-crafted" features [145]. This process of feature engineering is not only labor-intensive, but also underlines the weakness of many machine learning algorithms; their lack of ability for them to derive important factors from data [145, 146]. Moreover traditional approaches to feature engineering are highly dependent upon prior or commonsense knowledge. Deep Learning approaches are increasingly used to automatically engineer features for use in machine learning algorithms since they can uncover multiple levels of representation, with higher-level features representing more abstract aspects of underlying datasets [145].

With respect to inferring transportation mode with machine learning algorithms, hand-crafted aggregate trip features such as trip length, mean speed, mean acceleration, etc. are typically provided to classifier algorithms. As such, these methods typically fail to take account of the potentially rich data available on individual GPS points along a trip. In addition, some engineered features, such as maximum speed or average acceleration can be strongly collinear making it difficult for classifiers to use them to distinguish between modes. Deep learning approaches allow for the automated discovery of abstraction that can not only reduce the dependence of mode inference algorithms on feature engineering, or on prior knowledge of the modeller, but can also discover factors related to each GPS point that are usually overlooked by traditional machine learning algorithms. In this study, we use GPS-point information as input for Convolutional Neural Networks (CNN) to infer

the mode of transport. In addition, we take advantage of ensemble learning strategies to improve the prediction accuracy of CNN models.

This research uses data collected by the smartphone travel survey app, MTL Trajet, which is an instance of the smartphone travel survey platform, DataMobile/Itinerum [109]. MTL Trajet was released as part of a large-scale pilot study on the 17th of October 2016 in a study that lasted 30 days. Over 8,000 people participated in the study [18].

The rest of the paper is organized as follows: a background section describes previous work on transport mode inference. The methodology section sets out the framework of the CNN as well as of the data pre-processing and ensemble method configuration. The next section after that presents the results of the CNN and ensemble models on the MTL Trajet dataset. The last section concludes the paper.

4.2 Background

This section reviews previous research related to mode detection from smartphone data. We also describe the mathematical operations used in the Convolutional Neural Networks (CNN).

4.2.1 Mode Detection and Machine Learning

Mode detection has been done using various approaches in the literature, which include: rule-based, machine learning, and discrete choice approaches. Elhoushi et al. [147] have done a comprehensive and comparative literature review on travel mode detection using different sensors and different classification methods.

Tree-based ensemble classification algorithms have been used by Xiao et al. [40] to classify mode of transport. Their best ensemble method achieved a prediction accuracy of 90.77%. Wang et al. [56] develop a Random Forest classifier combined with a rule-based method to detect six modes of transportation using seven GPS-related features. Their method is able to detect more than 98% of subway trips with an overall accuracy of the other five modes classification as high as 93.11%. Assemi et al. [57] deployed a nested logit model with eight attributes to infer mode of transport from smartphone travel surveys, implemented in New Zealand and Australia. They have reported

an accuracy of 97% for New Zealand which includes data cleaning and 79.3% for Australia without any pre-processing.

Endo et al. [58] proposed a deep neural network approach to automatically extract high-level features. Their innovative approach converted raw GPS trajectories into a 2-D image structure and fed it as the input to a deep neural network. As an alternative to the RGB (Red, Green, Blue) values of an image pixel, stay time, i.e. the duration that a user stays in the location of the pixel, was used as the pixel value. They integrated hand-crafted features with the image-based feature. Eventually, they deployed traditional machine learning models, such as logistic regression and decision tree, to predict mode of transport. Although they devised an innovative idea to convert GPS trajectories into 2-D images, the pixel values only contained stay time without taking into account the spatiotemporal or motion characteristics, such as speed or acceleration, of the GPS trajectories. Their best models was able to detect the mode of transport with prediction accuracy of 67.9% on the GeoLife dataset and 83.2% on the Kanto Trajectories dataset.

More recently, Dabiri and Heaslip [39] used CNN models to train a mode detection classifier. They developed different architectures of CNN models on GPS trajectories, and finally combined their output via an ensemble method. Their ensemble library comprised seven CNN models. They took the average of the softmax class probabilities, predicted by each CNN model to generate the transportation label posteriors. Although the study carried out by Dabiri and Heaslip [39] used the CNN models, their study is different from our study in the following ways.

The segmentation method, the CNN architectures (explained in Section 4.3.1) and the survey size used in two studies are different. The latter is particularly important. The current study is based on a large-scale, real-world travel survey with about 8,000 participants, while the Dabiri and Heaslip study used trajectory data from 69 users. Furthermore, we have used a different ensemble method, i.e. a Random Forest model as a meta-learner explained in Section 4.3.1, which demonstrates better prediction performance over the ensemble method developed by Dabiri and Heaslip [39].

Apart from the features used in the mode inference literature, features used in other fields of study may contribute to transportation mode detection [39]. “Jerk” has been used in traffic safety analysis to identify critical traffic events. Jerk is defined as the derivative of acceleration, or the rate of change of acceleration over time [39]. GPS bearing rate has been used in driver behaviour profiling using

smartphone data [148]. Bearing rate is defined as the change of bearing between three consecutive GPS points [39] (Jerk and bearing rate are further explained in Section 4.3.1).

With respect to Convolutional Neural Networks, numerous architectures have been implemented in the broad Deep Learning literature, although their fundamental elements are very much alike. With respect to data dimensions, CNNs have been implemented on 1D, 2D and 3D data [146]. Since CNNs can learn features as well as estimate classifier coefficients, they are able to accomplish better prediction accuracy on large-scale datasets [146]. The success of CNNs in other fields of study encourages us to implement it as a classifier to infer mode of transport. The data used in this study represents a sequence of GPS points for each trip and can be considered as one-dimensional data. However, since CNNs typically use same-size input, we split the trip trajectories into same-length segments. The data preparation steps and modeling approaches have been explained in the next section.

4.2.2 Convolutional Neural Networks

CNN models are a class of neural networks suitable for processing grid-like topology data [33], which vary from 1D time-series data to 2D images. CNN models rely on affine transformation [33], which involves a vector of inputs being multiplied by a matrix (also called kernels, or filters) to produce an output. The multiplication by a matrix is referred to as convolution operation. Typically, a bias vector is added to the result of the matrix multiplication. Next, a non-linear function, called an activation function, is applied to the output of aforementioned operations. After the non-linear activation function, a pooling operation is typically applied. We briefly explain each of these stages below.

Generally, these mathematical operations, i.e. matrix multiplication, function activation and pooling, form one "hidden layer" of a CNN model. The output of each hidden layer of a CNN model can be fed as input into the next layer. The last layer of a CNN model produces class probabilities by applying an activation function, such as the sigmoid or softmax functions. Figure C.3a shows the general architecture of a CNN model.

Regardless of the dimensionality, the input data for a CNN model can be stored as fixed-sized multi-dimensional arrays (or tensors) [90]. Hence, GPS trajectories in our dataset need to be converted to

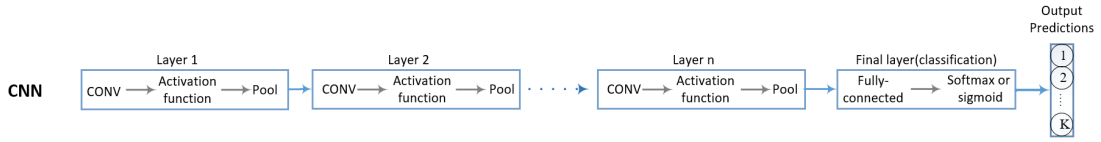


Figure 4.1: General architecture of Convolutional Neural Network. (CONV: Convolution)

fixed-sized arrays with different channels. This procedure is explained in Section 4.3.1.

4.2.3 Ensemble methods

Ensemble methods combine multiple classifiers and have been found to provide the possibility of higher accuracy results than a single classifier. Well-known ensemble techniques include boosting, bagging and stacking. Stacking combines the outputs of a set of base learners and lets another algorithm, referred to as the meta-learner, make the final predictions [149]. A super learner is another method that calculates the final predictions by finding the optimal weights of the base learners by minimizing a loss function based on the cross-validated output of the learners [149].

The most common ensemble method used for neural networks is average voting that generates posterior labels by calculating the average of the softmax class probabilities or predicted labels for all the base learners [149]. Majority voting is another ensemble approach that counts the predicted labels from all the base learners and reports the label with the maximum number of votes as the final prediction. Another approach is to calculate the optimal weights of individual base learners. The optimal weight of each base learner can be obtained by minimizing a loss function, i.e. the mean square error (MSE), given output of the base-learners. By minimizing the loss function, better performing classifiers are assigned larger weights. The final predictions of the ensemble are acquired by voting using the optimal weights of classifiers. Using a meta-learner is another ensemble approach that trains a learner, e.g. a Random Forest model, on the predicted class probabilities (or labels) of all base-learners to make the final prediction.

4.3 Methodology

In this section, we describe the data used, data pre-processing, the CNN architectures used, hyper-parameter value determination, and ensemble model configuration.

4.3.1 Data

Three types of data from the MTL Trajet dataset were used in this analysis; trip mode of frequent trips, coordinates data and user validated mode data. Upon installation of MTL Trajet, respondents were asked a series of questions. Two of these questions related to travel mode for trips from home to work or study location. For trips between these locations, respondents were asked what mode of travel they typically used, and whether or not other modes were ever used. Coordinates data contain respondent latitude and longitude obtained primarily through GPS. There are over 33 million location (primarily GPS) points in the MTL Trajet 2016 database. User validated mode data include information on trip mode, i.e. walk, bike, car, public transit, provided by respondents. When analyzing the data, it was found that mode data for trips between home, work and study for respondents that used one (and only one) mode was less noisy than mode data recorded from prompts. Hence, only validated trips from users who declared they used only one mode option to travel between home and work or home and school were used.

We used the rule-based trip-breaking algorithm developed in Patterson & Fitzsimmons [109] to detect start and end point of trips from raw GPS trajectories. The trip-breaking algorithm considered dwell time between GPS points as the most prominent criterion for detecting trips. The trip-breaking algorithm detects trip segments based on 3-minute gaps in data. Segments are then stitched back together while controlling for velocity and parameters relating to the public transit network (i.e. transit junctions and metro station location). For example, when two consecutive points are detected within 300m of a metro station (data collection is sparse when underground), and the gap is less than the maximum travel time by metro, the segments are joined as part of the same trip. Similarly, when two consecutive points fall within the same intersection with bus correspondences, additional time (10 minutes instead of 3 minutes gap) was allowed.

4.3.2 Data Preparation

We considered five channels for the input data for the CNN models: distance to previous point, speed, acceleration, jerk, and bearing rate, as recommended in the literature [39]. Speed is calculated using the distance between each two consecutive GPS points divided by their time interval.

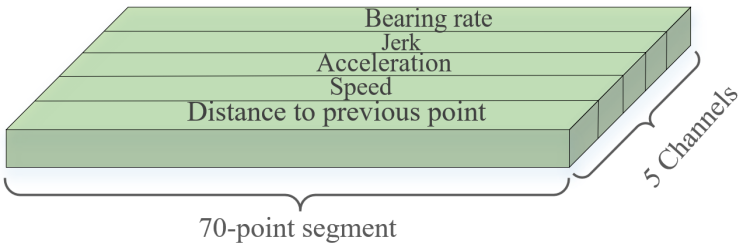


Figure 4.2: A 70-point Segment with 5 Channels.

Acceleration is also defined as the derivative of speed or the rate of change of speed over time. These characteristics constitute the channels of each segment as shown in Figure 4.2. These segments are fed as input to the CNN models. Jerk is defined as the rate of change in acceleration. Bearing is a term used typically in navigation and defined as the angle between the direction of vehicle to the destination and the magnetic north. Bearing is different from heading, as heading is the direction in which a vehicle is moving. However, like bearing, the heading is defined in degrees from magnetic north. Figure 4.3 demonstrates the bearing rate of a moving vehicle.

Bearing rate values vary across different transportation modes. For example, buses and cars usually do not experience sharp changes in steering angle while travelling between traffic lanes. As a result, their heading does not change regularly. However, pedestrians and bike riders more commonly experience changes in their heading. The formula to calculate the bearing (β) of two consecutive GPS points (p_1, p_2) is defined by following equations [39]:

$$\beta_1 = \arctan(X, Y) \tag{13}$$

where:

$$X = \cos(lat_{p_1}) * \sin(lat_{p_2}) - \sin(lat_{p_1}) * \cos(lat_{p_2}) * \cos(lon_{p_2} - lon_{p_1})$$

$$Y = \sin(lon_{p_2} - lon_{p_1}) * \cos(lat_{p_2})$$

The lon and lat represents the longitude and latitude of GPS points in radians. The bearing rate is

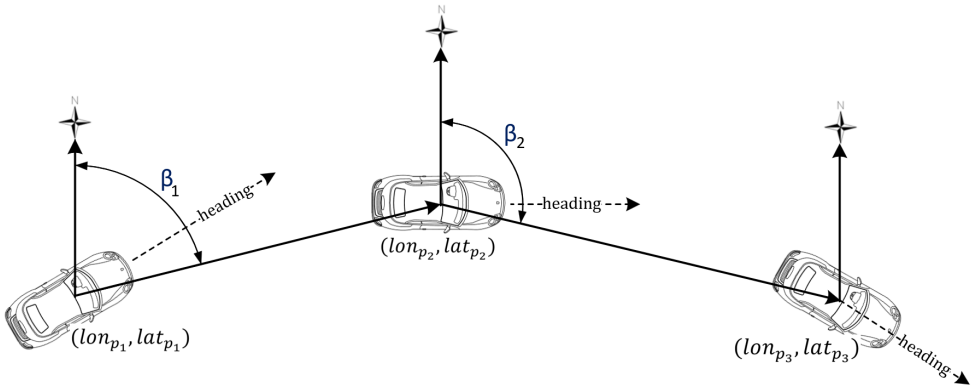


Figure 4.3: Bearing ($\beta_1 and \beta_2$) and Heading of a Moving Vehicle.

defined as the absolute change in the bearing of two consecutive GPS points and can be obtained by the following formula [39]:

$$Bearing\ Rate = \mid \beta_2 - \beta_1 \mid \tag{14}$$

Where β_1 and β_2 are the bearing of points p_1 and p_2 calculated by Equation 13. As shown in Figure 4.3, at least three consecutive GPS points are required to calculate the bearing rate, since the bearing of a GPS point is calculated according to its following point. For example, as shown in Figure 4.3 the bearing of vehicle at the first position, i.e. p_1, is calculated according to p_2, and the bearing of the vehicle at p_2 is calculated according to p_3, as in above formulas.

The input data of the CNN models needs to be of fixed size. As the number of GPS points along trips vary, we tried to split trips into m-point segments, where m was the average (70 points) or median (120 points) number of points for trips in the dataset. After examining both 70 and 120 point segments we found the best performance of CNN with seventy-point (average length) segments. For those segments with the number of points less than 70, the remainder of the segment was padded with zeroes. The resulting dataset consisted of 3,845; 8,515; 7,415 and 15,275 walk, bike, transit and car segments, respectively.

We also implemented other data pre-processing steps to remove errors from GPS trajectories. We removed trajectories with less than 10 GPS points. Furthermore, we only considered segments from

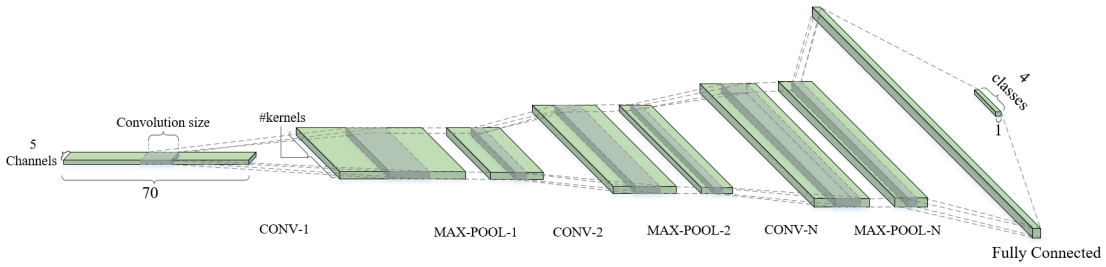


Figure 4.4: Convolutional Neural Network Architecture with N convolution layers and one Fully-connected layer.

those respondents who had validated at least three trips during the survey. Also, restrictions on speed and acceleration between two consecutive GPS points was applied as suggested by Dabiri and Heaslip [39].

4.3.3 CNN Architecture

The overall architecture of the CNNs used in the current study is illustrated in Figure 4.4. A typical CNN architecture consists of three layers: convolutional, pooling and fully-connected layers. Our model begins with segments of size 70×5 that contains the 70 GPS points and 5 channels. The convolutional layers of the CNN are formed by a convolution operation followed by a max-pooling operation. The Leaky Rectified Linear Unit (Leaky ReLU) activation function is applied to the output of every convolutional layer. The last layer is fully-connected. The output of the last fully-connected layer is fed to a 4-way softmax that produces a distribution over the 4 modes of transportation.

Six different CNN architectures, shown in Table 4.1, were evaluated. Model A is a basic model with only one convolutional and one fully-connected layer. Model B consists of four convolutional and one fully-connected layer with convolutional and pooling kernels of size 8×1. Model C uses a deeper architecture consisting of 20 convolutional layers and one fully connected layer. It has the same number of kernels and kernel sizes for convolution as Model B, but with a max-pooling layer of size 4×1. We make a deeper architecture in Model C by repeating the layers five times. We implement Model C (and also Model E, as explained below) to evaluate the performance of wide versus deep architectures for CNN models on our dataset.

Model D is a wide neural network, i.e. it uses a large number of convolutional kernels relative to the number of layers. The architecture consists of 5 convolutional layers followed by one fully connected layer. The number of kernels in each layer is similar to the ones suggested by Krizhevsky [150]. The first convolutional layer consists of 96 kernels of size 8×5, which are applied with a stride of 1 on the 70×5 input segment. The second convolutional layer takes the output of the first convolutional layer as input and has 256 kernels of size 8×96. There are 384 kernels of size 8×256 in the third and 384 kernels of size 8×384 in the fourth convolutional layer. Also, the fifth convolutional layer consists of 256 kernels of size 8×384. Each convolutional layer is followed by a Leaky ReLU activation function and a max-pooling layer of size 8×1. To keep the output size the same as the input size we use the *same* padding for both convolutional and max-pooling kernels [90]. Also, we use stride 1 for both convolutional and max-pooling layers to allow a high spatial resolution of GPS points.

Model E uses a deeper architecture, relative to Model D, consisting of 20 convolutional layers and one fully connected layer. The architecture of Model E is similar to the architecture of Model D in terms of number of kernels and kernel sizes. To reach a deeper architecture, each layer is repeated four times. We use a 2×1 max-pooling layer after each convolutional layer with unit stride and the *same* padding. Model F contains a larger number of kernels than the previous models. We implement Model F to assess how far the larger number of kernels can improve the prediction accuracy of a CNN model. The model consists of 6 convolutional layers with 128, 256, 512, 1024, 1024 and 512 kernels. The convolutional and pooling kernels are of size 8×1. Finally, we applied the fully-connected layer. For all the models, to avoid over-fitting we applied the Dropout method before the fully-connected layer, and set each hidden neuron to zero with a probability of 0.5.

We tested different values for the size of convolutional (kernel) and pooling size, from 2 to 16. We found lower prediction accuracies when the kernel or pooling sizes were set to 16. However, with respect to kernel size, all the models with kernel size of 8 showed higher prediction accuracy. This was not the same for pooling size, where as in deeper models, such as models B and E, the prediction accuracies were better when we set the pooling size to 4 in Model B and 2 in Model E in Table 4.1.

Table 4.1: CNN Model Architectures

Model A	Model B	Model C	Model D	Model E	Model F
2 layers	5 layers	21 layers	6 layers	21 layers	7 Layers
Input segment					
conv 8-4 maxpool 4	conv 8-4 maxpool 8	$5 \times \begin{pmatrix} conv8-4 \\ maxpool4 \end{pmatrix}$	conv 8-96 maxpool 8	$4 \times \begin{pmatrix} conv8-96 \\ maxpool2 \end{pmatrix}$	conv 8-128 maxpool8
	conv 8-8 maxpool 8	$5 \times \begin{pmatrix} conv8-8 \\ maxpool4 \end{pmatrix}$	conv8-256 maxpool 8	$4 \times \begin{pmatrix} conv8-256 \\ maxpool2 \end{pmatrix}$	conv8-256 maxpool 8
	conv 8-16 maxpool 8	$5 \times \begin{pmatrix} conv8-16 \\ maxpool4 \end{pmatrix}$	conv8-384 maxpool 8	$4 \times \begin{pmatrix} conv8-384 \\ maxpool2 \end{pmatrix}$	conv8-512 maxpool 8
	conv 8-32 maxpool 8	$5 \times \begin{pmatrix} conv8-32 \\ maxpool4 \end{pmatrix}$	conv8-384 maxpool 8	$4 \times \begin{pmatrix} conv8-384 \\ maxpool2 \end{pmatrix}$	conv8-1024 maxpool 8
			conv8-256 maxpool 8	$4 \times \begin{pmatrix} conv8-256 \\ maxpool2 \end{pmatrix}$	conv8-1024 maxpool 8
					conv8-512 maxpool 8
Dropout					
Fully Connected					

4.3.4 Ensemble Model

In addition, we apply four methods to combine the predictions of CNN models: (a) average voting (b) majority voting (c) optimal weights of individual base learners and (d) a random-forest model as meta learner. For the third method, to find the optimal weight of each base-learner, we minimize a loss function, i.e. the mean square error (MSE), given output of the base-learners. By minimizing the loss function, better performing classifiers are assigned larger weights. The final predictions of the ensemble are acquired by voting using the optimal weights of classifiers.

With respect to the fourth method, the predictions of base learners are fed to a Random Forest classifier to make the final predictions. The Random Forest model was built using 800 trees. Also, we allowed up to 8 attributes to be randomly sampled at each split.

In this study a series of CNN models as well as an ensemble method have been used. We used ensemble models to improve the prediction accuracy of the CNN models. In addition, we tested four different ensemble approaches for achieving the highest prediction accuracy. We refer to the CNN models in the ensemble library as level-0 models and the methods to make the final predictions as level-1 models. Level-0 includes models A-F (Table 4.1) as well as other base learners developed based on different CNN architecture types and hyper-parameter values, shown in Table 4.2. Level-0 contains 75 base learners. Finally, each method in level-1 is applied on the output of Level-0 models

Table 4.2: Hyper-parameter values used in Ensemble

Name	Hyper-parameter value
Convolutional layer	Num. of layers: 6, 7, 11, 21 Num. of kernels: 2,4,8,16,32,98,128,256,384,512,1024 Kernel size: 2, 4, 8 Stride size: 1, 2 Stride type:SAME Activation function: Leaky ReLU
Max pooling layer	Pooling size: 2, 4, 8 Stride size: 1, 2 Stride type:SAME
Output layer	Activation: Softmax
Optimization method	Adam optimizer with learning rate = 1e-4
Batch size	16
Number of epochs	20, 50, 100

to predict the final labels.

4.4 Results and Discussion

The CNN architectures were programmed in Tensorflow with GPU support. We used randomized stratified sampling to ensure that all modes were represented equally in the training and test sets.

4.4.1 CNN models

In our first experiment we assess different CNN architectures in terms of: number of layers and kernels, kernel and pooling size, stride size, activation function, optimization method and batch size. We tested both Gradient Descent and Adam optimization algorithms, and found that the Adam algorithm compares favorably to Gradient Descent. For Adam, we used a decay learning rate equal to 0.95 and set the starting learning rate to 0.01. We also tested the fixed learning rate equal to $1e-4$ and found there is not any significant difference between the results of models with decay or constant learning rates. Hence, we set the learning rate to $1e-4$ for all the models. We did our first experiments with number of epochs equal to 20 and 50. To avoid over-fitting we used dropout and early stopping methods [151]. With respect to batch size, we tested sizes of 16, 32, 64 and 128, and got better results with a batch size of 16. We also tested "Tanh", "Relu" and "Leaky Relu" activation functions and achieved better results with Leaky Relu function.

Table 4.3: Test Accuracy of the Models

Model type	Model A	Model B	Model C	Model D	Model E	Model F	Ensemble (Average Voting)	Ensemble (majority voting)	Ensemble (optimal weights)	Ensemble (RF as meta-learner)
Accuracy(%)	72.5	75.6	75.7	81.3	80.6	80.1	85.0	89.1	89.4	91.8

Table 4.4: confusion matrix analysis (ensemble with RF as meta-learner)

Mode of Transport	Walk	Bike	Transit	Car	Precision(%)	Recall (%)	F-score(%)
Walk	708	32	29	0	91.1	92.0	91.6
Bike	32	1573	60	38	91.8	92.3	92.1
Transit	30	66	1296	91	84.8	87.4	86.1
Car	7	42	142	2864	95.7	93.8	94.7

Table 4.3 shows the results of six CNN models as well as the ensemble models. Model A is a basic model and shows that a CNN with only one convolutional layer is able to predict the mode of transport with 72.5% accuracy. Models B and C have different numbers of layers, i.e. 6 and 21 layers respectively, and possess the same number of kernels as well as kernel and pooling size. The prediction accuracy of Models B and C is almost identical, although model C is marginally better than model B.

Models D-F comprise higher numbers of kernels in each layer than the previous models and display higher prediction accuracies. Models D-F show prediction accuracies of 81.3%, 80.6% and 80.1%, respectively. The trade-offs between the depth (number of layers) and width (number of kernels) of neural networks have been investigated by some researchers [152]. In fact, deeper networks can show better results than shallower networks [152]. However, as He et al. [152] have mentioned, at some point, the error in prediction accuracy of CNN models not only gets saturated, but gets worse as the models get deeper. We see such a pattern when comparing models E and F with model D that while deeper has lower prediction accuracy. Similar observations have been reported by Dabiri and Heaslip [39]; they also found that a shallower network performed better than a network with deeper configuration.

4.4.2 Ensemble model Results

Table 4.3 illustrates the prediction accuracy of ensemble models with different combining methods. The first method (average voting) leads to a higher accuracy than even the best CNN model, i.e. Model D. A better performance (89.4% prediction accuracy) is obtained using the majority voting

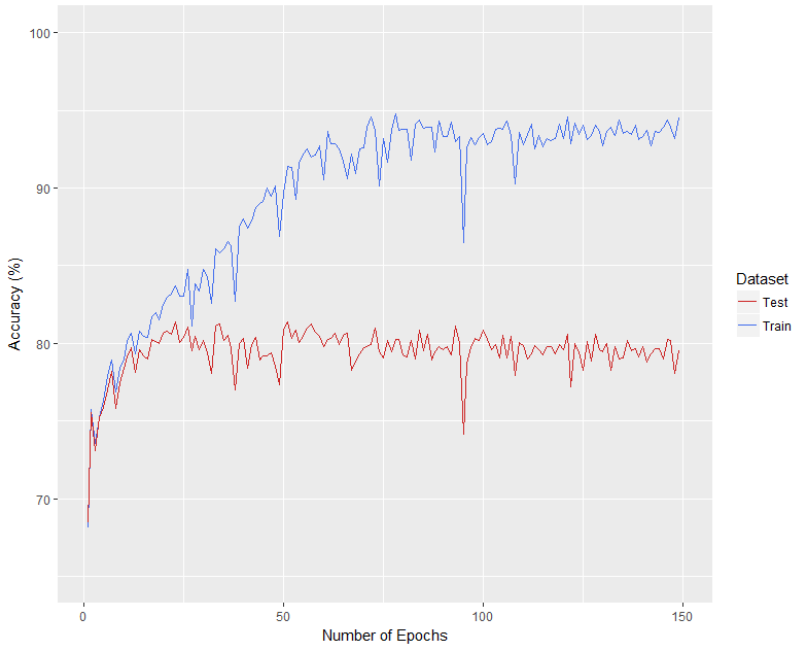


Figure 4.5: Train and Test Accuracy for Different Numbers of Epochs (Model D).

method. Finding the optimal weights of base CNN models leads to the same prediction accuracy as the majority voting method. The highest prediction accuracy is obtained when using the Random Forest model as meta-learner (roughly 91.8%).

Table 3.6 shows the confusion matrix of the ensemble method with RF as meta-learner, using a 5-fold cross validation. Walk, bike, transit and car segments are predicted with 91.1%, 91.8%, 84.8%, 95.7% precision, respectively. Recall and F-score are also reported with the highest and lowest values for car and transit. Car trips are easier to infer due to their higher speed and acceleration. However, bus trips share some similarities with car trips, especially in dense urban areas where cars and buses experience the same traffic conditions.

Figure 4.5 shows the accuracy of Model D on train and test . The accuracy on train dataset increases with the number of epochs and the test accuracy remains constant after around 50 epochs. Apart from what was mentioned above, another reason that accuracy on test dataset in neural networks does not increase while the train accuracy goes higher without any over-fitting is the insufficient number of samples in the training data [39]. Hence, the inclusion of more transit trips to the dataset could possibly enhance the prediction accuracy of the CNN models.

4.4.3 Comparison with classical machine learning models and previous studies

To assess the prediction accuracy of the CNN models and ensemble methods in this study, we developed a Decision Tree (DT) as a classical machine learning algorithm as well as a Random Forest (RF) model, which is an ensemble machine learning approach. We used several hand-crafted features, as explained in [153], including average speed (km/h), 85th percentile speed, maximum and minimum acceleration, travel time, etc. Like CNN models, there are hyper-parameters for DT and RF. The tree depth is controlled by the required minimum number of observations in each node. We tested different values, ranging from 1 to 40 for this parameter. The highest prediction accuracy on test datasets was acquired with the required minimum number of observations in each node equal to 20. With respect to the RF model we tried different numbers of trees in each forest, ranging from 100 to 2000. We found no improvement in prediction accuracy of RF models with number of trees higher than 1000. The resulting DT reached the prediction accuracy of 73.4% for detecting mode of transport. Also, the RF model, reached a predicting accuracy of 86.8%. Comparing these results with the results of CNN models and ensemble models shows that the ensemble methods, either a Random Forest (ensemble of decision trees) model or an ensemble of CNN models, outperform the single learners, such as the DT or single CNN models in our study. In addition, the CNN model in some cases, such as Models A-C in Table 4.3, demonstrates a prediction accuracy close to the DT. However, with wider or deeper CNN architectures, such as Models D-F, we can reach higher accuracy prediction performance.

It has been said that comparing prediction accuracy rates from different studies can be misleading because of differences in sample sizes, number of classes, and the quality of data across studies. Bearing these considerations in mind, our findings compare favorably with previously obtained results in the literature. Dabiri and Heaslip [39] have reported a test accuracy of 84.8% for an ensemble Convolutional Neural Network. Their best model, is able to predict walk, bike, bus, driving and train, with 81.6%, 90.3%, 80.7%, 86.6%, 92.3% precision, respectively. Also, Zheng et al. [64] have reported accuracies of 89%, 86%, 66% and 65% for walk, driving, bus and bike modes, respectively. Their decision tree-based inference model, results in an overall accuracy of 76.2%. Both the aforementioned studies [39, 64] used GPS data from mobile phones, similar to

that collected in the MTL Trajet data, although they had much smaller sample sizes of 69 and 65 persons, respectively. Our best ensemble model shows an overall accuracy of 91.8% and is able to predict walk, bike, public transit, car with 91.1%, 91.8%, 85.8%, 95.7% precision based on data from over 8,000 users. The results demonstrate that overall accuracy of our best model is superior to the overall accuracy of both aforementioned comparable studies.

Also, Gonzalez et al. [69] produced precision rates of 92% for car trips and 81% for bus trips. Bantis and Haworth [118] developed several models to detect three modes of transport (walk, bus/car and train) using stationary points from GPS, accelerometer data as well as user characteristics. Their training data contained trips from 5 individuals. They proposed a hierarchical dynamic Bayesian network and compared the results against Random Forest, SVM, and Multilayer Perceptron classifiers. The accuracy of their proposed model was 90%, while it could predict Stationary status as well as Walk, Bus/Car and Rail trips with 94%, 65%, 88%, 91% precision, respectively. Although their categories are different from ours and the sample size of the two studies are not similar, our best ensemble model predicts walking trips with much higher precision, i.e. 91.1% vs. 65%. Moreover, our best ensemble model predicts car trips with 95.7% precision, higher than all aforementioned studies.

4.5 Conclusion

We developed a series of Convolutional Neural Networks augmented by different ensemble methods to infer travel mode from trajectories gathered by a large-scale smartphone travel survey. The raw trajectories of travellers were tailored in a manner that allowed feeding them as an input layer to a CNN. Each trip was segmented into fixed sized segments with five channels. The channels include "distance to previous point", "speed", "acceleration", "jerk", and "heading rate". We investigated different CNN architectures and combined their results via different ensemble methods to obtain the highest prediction accuracy. Our ensemble library was composed of a series of CNN models with different hyper-parameter values and CNN architectures. Afterwards, we combined the results of CNN models with "average voting," "majority voting" and predicting the optimal weight of each

classifier. Furthermore, we exploited the ensemble library by deploying Random Forest as a meta-learner. We find that all the ensemble methods outperform the individual CNN models. Moreover, the ensemble method with Random Forest as meta-learner shows an accuracy of 91.8% which surpasses the other three methods. Finally, the "majority voting" and "optimal weight" combination methods result in similar prediction accuracy rates around 89%.

This study contributes to the mode detection literature and the ITS community by using a meta-learner classifier (Random Forest in our case) to aggregate the output of several base-learners (CNN models). While an approach using a meta learner ensemble, i.e. applying a learner over the output of a set of CNN base-learners, has been used in some applications in other fields [149], to the best of our knowledge the approach has not been used in previous mode detection studies. Furthermore, the data set used in this study has been collected as part of a large-scale travel survey, that shows the capability of such smartphone-based travel surveys as a complementary (or even a replacement) surveying tool to the current real-world household travel surveys.

In addition, considering that our study has been developed on trajectories gathered as part of a large-scale and real world travel survey, we believe that smartphone travel surveys have the potential to replace traditional travel surveys, such as face-to-face or computer assisted telephone interviewing(CATI), although there are still many hurdles to overcome. One of these hurdles is the quality of the labelled trajectories. The labelled trajectories produced by large-scale smartphone travel surveys may not show the same quality as labelled trajectories from small or researcher-collected smartphone travel surveys. Improving the trajectory collection and validation techniques for large-scale travel surveys should be a focus of future studies while deploying state-of-the-art classification techniques. Furthermore, as the need for labelled data is a requirement of supervised machine learning models used in the literature and the current study, using semi-supervised or unsupervised models may help to reduce or eliminate the need for labelled data.

Acknowledgments

This research is based upon work supported by the Social Sciences and Humanities Research Council of Canada as well as "MTL Trajet" research contract with Ville de Montréal. Also, this research

was enabled in part by support provided by WestGrid Computing Facilities and Compute Canada.

Chapter 5

Semi-supervised Generative Adversarial Networks to Infer Transportation Mode from GPS Trajectories

Preamble

This chapter introduces Generative Adversarial Networks (GANs) as a semi-supervised approach to detect mode of transport. The models developed in Chapters 3 and 4 were supervised learners. However, as mentioned in the literature review, Section 2.1, the are very few studies addressing the application of semi-supervised learning approaches in trip information inference from trajectory data. The GANs model developed in this chapter is a single-task learner and uses the point-based features of GPS trajectory to identify mode of transport.

The semi-supervised approach in this chapter, is a generative model, which produces synthesized trajectories similar to the real trajectories in the MTL Trajet dataset, and attempts to enhance the performance of mode inference learner. The learner (discriminator) is this chapter has been developed based on the architecture of the CNN learner in Chapter 4.

Abstract

This study experiments with the use of adversarial networks to classify travel mode based on one-dimensional smartphone trajectory data. We use data from a large-scale smartphone travel survey in Montreal, Canada. We convert GPS trajectories into fixed-sized segments with five channels (or variables). We develop different GANs architectures and compare their prediction results with Convolutional Neural Networks (CNNs). The best semi-supervised GANs model led to a prediction accuracy of 83.4%, while the best CNN model was able to achieve the prediction accuracy of 81.3%. The results compare favorably with previous studies, especially when taking the large, real-world nature of the dataset into account. The developed semi-supervised GANs models share the same architectural innovations used in the image recognition literature, that we show can be used in travel information inference from smartphone travel survey data, not only to generate more labeled samples, but also to improve the prediction performance of the classifier . Future work will allow exploration of better performing models either with more channels and/or improved architectures.

5.1 Introduction

Over the past few decades, transportation planners have used travel surveys to collect travel demand behavior in transportation planning and demand modeling. Traditional survey methods, such as face-to-face interviews or Computer Assisted Telephone Interviewing (CATI) suffer from some disadvantages including considerable implementation costs and decreasing response rates. Other shortcomings, such as burden on respondents and respondent fatigue or forgetfulness, which cause inaccuracies and error in gathered data, have been identified in many studies [21, 22]. With the ubiquity of smartphones and advances in their technology, such as being equipped with different sensors such as GPS and accelerometers, researchers are beginning to employ smartphones as a travel data gathering tool. Inferring transport mode from GPS traces is one of the important steps in deriving trip information from smartphone-based travel surveys.

Different approaches in the literature have been used to infer mode of transport from GPS traces, such as rule-based algorithms [46], machine learning classifiers [59, 117], deep learning [39] and discrete choice modeling [50]. Machine/deep learning methods can be divided into three large categories of learning approaches: supervised-, semi-supervised- and unsupervised learning (explained in the next section). Most of the literature concerning machine learning and information inference from GPS and smartphone data has used supervised learning approaches. Moreover, deep learning (supervised) algorithms typically show their best performance on extremely large labeled datasets [102]. One of the main challenges of these methods is thus obtaining (or gaining access to) such datasets. Semi-supervised learning is one flexible strategy to reduce the required number of labeled examples by studying large unlabeled data sets, which are easier to obtain [102].

Recently, Generative Adversarial Networks (GANs) [95] have shown promising results in image and language processing. In the GANs framework, a generative model is set against an adversarial, or a "discriminative" model, that learns to distinguish between the observations produced by a "generative model" and real data observations [95]. While the GANs framework has mainly been used for generating samples, it can be trained for semi-supervised learning, where the labels for a considerable part of examples are missing. One example is CatGANs [100], which successfully trained a discriminator classifier on unlabeled and partially labeled data.

This study investigates the development of a semi-supervised GANs to infer mode of transport from the GPS traces of travelers. The paper is organized as follows: a background section describes the literature related to transportation mode inference and generative adversarial networks. Next we describe the methodology used to train a semi-supervised GANs framework to detect mode of transport from smartphone data. Also, the methodology section introduces the data processing steps. Afterwards, the results section describes the developed models and their prediction results. Finally, we present our conclusions and future directions.

5.2 Background

As mentioned in the previous section, there are three basic approaches for training in machine/deep [154]: supervised, semi-supervised and unsupervised approach. In supervised learning, a set of labeled training data is used to infer a function that maps an input to an output based on input-output pairs. Semi-supervised learning is also a supervised learning class but where only a subset of training data is labeled. Semi-supervised learning methods can still take advantage of unlabeled data for training. In unsupervised learning, all training data are unlabeled. Unsupervised learning infers a function that describes the structure of data and groups them. This section, briefly describe the unsupervised and semi-supervised GANs approaches. Before that, we briefly review previous research related to machine/deep learning used in mode detection from smartphone data.

5.2.1 Mode Detection and Machine Learning

Mode detection methods have been applied on various data sources including GPS trajectories [37, 39, 153], accelerometer data from smartphones [44], and Wi-Fi signals [155]. Endo et al. [58] attempted to use deep neural networks to automatically extract high-level features. They introduced an innovative idea to convert the raw GPS trajectory points into a 2-D image structure as the input into the deep neural network model. Instead of RGB (Red, Green, Blue) values of an image pixel, they equivalently consider the duration time that a user stays in the location of the pixel. They deployed traditional classifiers, such as logistic regression and decision tree, to predict transportation mode. Their best models were able to detect the travel mode with prediction accuracy of 67.9%.

Assemi et al. [57] developed a nested logit model to infer travel mode from smartphone travel surveys, using eight attributes, i.e. speed distribution skewness, percentage of total trip travel time with the speed ranging between 2 and 8 m/s, percentage of total trip travel time with the speed ranging between 8 and 15 m/s, maximum speed, 95 percentile of acceleration, acceleration variance, maximum acceleration, and ratio of direct distance to travelled distance between origin and destination. Their model can predict travel mode on GPS data gathered in Australia with 79.3% of accuracy. They have also reported an accuracy of 97% based on the GPS data gathered in New Zealand. However, they have mentioned that the high accuracy has been gained after undergoing an excessive data pre-processing step. Wang et al. [123] used CNN models for deep feature learning. They categorized the attributes into two classes: the point-level features and the trajectory-level features. They selected speed, heading change, time interval, and distance as the point-level handcrafted features. They considered average speed, variance of speed, heading change rate, stop rate, and speed change rate as the trajectory-level handcrafted features. Finally, they fed both type of features into a deep neural network classifier.

More recently, Dabiri and Heaslip [39] used Convolutional Neural Network (CNN) to train a mode detection model with an accuracy of 79.8%. They implemented different CNN models on GPS trajectories, and finally combined the output of the CNN models via an ensemble method. Their ensemble library was made up of seven CNN models from which they took the average of softmax class probabilities, predicted by each CNN model, to generate the transportation label posteriors.

5.2.2 Generative Adversarial Networks

GANs were introduced primarily as unsupervised learners that benefit from setting up a supervised learning framework. However, besides unsupervised learning, GANs are also able to undertake semi-supervised learning. Semi-supervised GANs have been implemented via three different approaches in the literature. Class-conditional GANs were first introduced by Mirza and Osindero [101]. As Mirza and Osindero [101] have explained, the unconditioned (traditional) generator in GANs has no control on the class labels of the data being generated. Mirza and Osindero [101], proposed a model able to direct the generating process by conditioning the generator on additional information [101]. They explain how GANs are able to do multi-class labeling by conditioning the

generator on class labels. Their proposed model can generate samples for each class. For example, in image classification problems, you can ask Conditional GANs for a "horse" class, and it will produce a picture of a horse.

The second approach for semi-supervised GANs enables it to predict K different output classes. In this approach, instead of predicting one "real" class and one "fake" class, the GANs predicts K different real classes. Fake (generated) data should result in the GANs being not confident about which class to output. This approach was first developed by Springenberg [100] in the CatGAN mode.

In the third approach, the GANs model outputs K+1 different classes. This semi-supervised GANs approach was first introduced by Salimans et al. [99] and Odenda [156]. Salimans et al. [99] added samples from the GANs generator G to their dataset, as a new "generated" class, and then used a standard classifier to do the semi-supervised learning. So, if we consider a standard classifier that classifies a data point x into one of K possible classes, the semi-supervised classifier will classify the data point into K+1 classes, with the K+1th class containing the observations "generated" by a GAN [99].

In this research we use the third approach proposed by Salimans et al. [99], because our main objectives of using GANs is to classify the mode of transport and take advantage of generated samples to improve the accuracy of classifier. The first approach, i.e. the conditional GANs, is more relevant for generating samples for each specific class, not classification. The second approach is able to classify the mode of transport, but does not directly benefit from the generated sample, as the classifier only classifies the K real classes. But, the third approach, the one that we aim to use, can do the classification and directly gaining advantage of generated samples (by considering them as "K+1"th class) to enhance the prediction performance.

While GANs is mainly used as a generative model to generate samples through labeling the data into "fake" and "real" [102], some studies have extended GANs for multi-class labeling. However, almost all implementations of GANs models have been conducted in image processing or speech recognition studies. To the best of our knowledge, analyzing GPS trajectories with semi-supervised GANs models to detect the travel mode has not been done in the literature yet. Investigating how semi-supervised GANs can benefit to mode detection by using generated samples is a question

worth exploring.

5.3 Research Design and Methodology

The most well-known GANs architecture is the Deep Convolutional GAN (DCGAN) [157]. Most GANs use at least some of the architectural innovations proposed in the DCGAN architecture. This section describes the architectures of semi-supervised DCGANs models that are developed. Moreover, a data section describes the data and the pre-processing steps implemented on the trajectory data.

5.3.1 Data

The entire MTL Trajet dataset comprises more than 33 million GPS points from over 8,000 respondents. To identify trips and segments, the trip-breaking algorithm, which is a rule-based algorithm, developed in Patterson & Fitzsimmons [109] was used. The algorithm recognizes segments, whenever it detects a 3-minute gaps in data while checking speed and the public transit network locational data. After implementing the trip-breaking algorithm on the GPS points, 623,718 trips were detected, among which 102,904 trips were validated by respondents. Afterwards, we applied several pre-processing and segmentation steps explained in the following.

Data Preparation

The input layer to the CNN or DCGAN models need to be of fixed size [39, 154], due to the mathematical procedures and calculations [90], such as fixed-size kernel and stride in each layer of a CNN model. However, the number of GPS points along a trip are not equal for all the trips in the MTL Trajet dataset. Hence, we split the detected trips into fixed size segments. We examined the average (70 points) and median (120 points) number of points for the length of the fixed size segments. Following testing both 70 and 120 point segments, we observed stronger performance of neural network models on seventy-point segments. We padded with zero when the segment size was less than seventy points.

With respect to the labeling method, the smartphone app prompted travelers to validate their travel

mode whenever a stop was detected throughout their movements during the day. In addition, respondents were asked to declare their travel mode to main destinations by answering a questionnaire when installing the MTL Trajet application. In particular, respondents were asked to reveal the location of home, school and work and home, along with the transportation mode(s) used to travel to these locations. We considered only the validated trips of those users who stated that they used only one mode option to make a trip between home and work/school. The final dataset thereby consists of 3845, 8515, 7415 and 15275 walk, bike, transit and car segments, respectively.

We calculated five characteristics for each point along a segment: "distance to previous point", "speed", "acceleration", "jerk", and "bearing rate", which have been used in several studies [39, 158, 159]. These characteristics are shown as channels of each segment in Figure 5.1. Afterwards, these segments were fed as input to the discriminator of the semi-supervised DCGANs model. Jerk describes the rate of change of acceleration. Bearing, as shown in Figure 5.2, is defined as the angle between magnetic north and the direction of a point to its consecutive next point.

We can obtain the bearing (β) by applying the following equations on any two consecutive GPS points, i.e. p_1,and p_2 [39]:

$$\beta_1 = \arctan(X, Y) \tag{15}$$

where:

$$X = \cos(lat_{p_1}) * \sin(lat_{p_2}) - \sin(lat_{p_1})) * \cos(lat_{p_2}) * \cos(lon_{p_2} - lon_{p_1}) \tag{16}$$

$$Y = \sin(lon_{p_2} - lon_{p_1}) * \cos(lat_{p_2}) \tag{17}$$

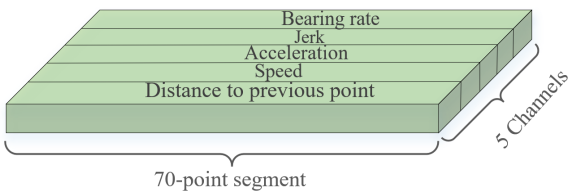


Figure 5.1: A 70-point Segment with 5 Channels.

The lon and lat are the longitude and latitude of GPS points in radians. The bearing rate is the absolute change in the bearing of two consecutive GPS points and calculated by the following formula [39]:

$$Bearing\ Rate = \mid \beta_2 - \beta_1 \mid \tag{18}$$

Where β_1 and β_2 are the bearing of points p_1 and p_2 calculated by Equations 15-17. From this formula, the bearing rate is equal to the absolute change in the bearing of two consecutive points. Also, it is necessary to have at least three consecutive points to measure the bearing rate (as demonstrated in Figure5.2), because the bearing of each point is calculated according to its following point.

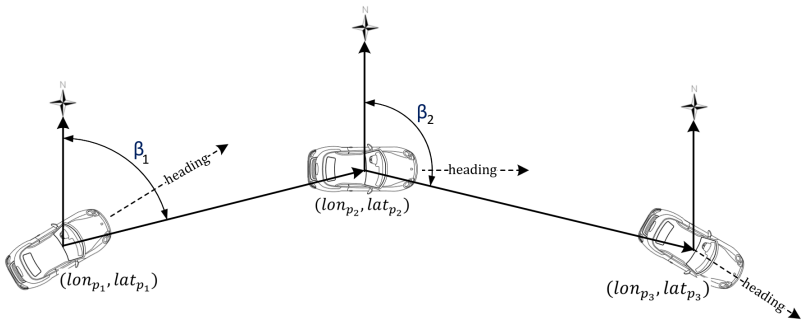


Figure 5.2: Bearing ($\beta_1 and \beta_2$) and Heading of a Moving Vehicle.

5.3.2 Details of GANs Training

We trained the semi-supervised DCGANs on the MTL Trajet dataset. Hyperparameter optimization is very important in training GANs. We trained models based on hyperparameters values previously examined in the literature and random values that we thought were useful to test. We tested different numbers of layers for both discriminators and generators from 2, 3, 4 and 12. Also, with respect to the kernel size, different values were tested; 4,8,16, and 32. The number of filters per discriminator and generator layer were tested based on the following values: 8, 16, 32, 64, 128, 256 and 512.
We trained different models with different batch sizes of 16, 32, 64 and 128. The models with a batch size of 16 demonstrated better results. With respect to the activation function, we tested

'Relu', 'Leaky Relu' and 'Tanh' functions and got higher prediction accuracies with the Leaky Relu function.

We applied one-sided label smoothing [99] to the positive labels of the generator. Label smoothing is a technique that replaces the 0 and 1 labels for a classifier with smoothed values like 0.9 or 0.1. This technique has been shown to enhance the performance of neural networks while facing adversarial examples [160]. We used the Adam optimizer for training both generator and discriminator. In addition, gradient clipping was used to keep the training procedure in steady state. Gradient clipping prevents the norm of the gradient from exceeding a given value [33]. In the next section the results of best trained GANs architecture for detecting GPS trajectories have been presented. Moreover, we trained different CNN architectures and presented the best models as the baseline models to be compared with GANs.

5.3.3 The Model Architecture

Table5.1a shows the architecture of different CNN and DCGANs models developed in this paper. Models A, B and C, in Table 5.1a, are the CNN models and Models E and F are the semi-supervised GANs models.

The generator and discriminator architecture of Model E are illustrated in Figure 5.3a and 5.3b, respectively. We used convolutional layers for training the discriminator in the DCGANs framework. There are usually three types of layers in a typical CNN architecture: convolutional, pooling and fully-connected layers. We did not use any pooling layer in the DCGAN discriminator as suggested by Radford et al. [157]. Our discriminator, in Model E, begins with segments of size 70×5 that contains the 70 GPS points and 5 channels. Afterwards, three convolution operations with kernel size of 8 and stride of 2 are used. The number of kernels in each layer is equal to 128, 256, and 512, respectively. The first convolution operation converts the input segment into a 35×128 output. The output of the second and third convolutions are of size 18×256 and 9×512, respectively. Afterwards, a fully connected operation is applied to the output of the third convolution layer and produces a fully-connected layer of size 1×4608. The output of the last fully-connected layer is fed to a 5-way softmax that produces a distribution over the 4 modes of transportation plus the fake class, in other words the discriminator has K+1 output units corresponding to [Class-1, Class-2,...,

(a) Architectures of Convolutional Neural Networks and Semi-supervised GANS

Model A	Model B	Model C	Model D		Model E	
CNN	CNN	CNN	Semi-supervised GANS		Semi-supervised GANS	
			Generator	Discriminator	Generator	Discriminator
Input $[70 \times 5]$	Input $[70 \times 5]$	Input $[70 \times 5]$	Z [100]	Input $[70 \times 5]$	Z [100]	Input $[70 \times 5]$
CONV8-32 MAXPOOL8	CONV8-128 MAXPOOL8	CONV8-96 MAXPOOL8	Projection& reshape	CONV8-32	Projection& reshape	CONV8-128
CONV8-64 MAXPOOL8	CONV8-256 MAXPOOL8	CONV8-256 MAXPOOL8	FS-CONV8-128	CONV8-64	FS-CONV8-512	CONV8-256
CONV8-128 MAXPOOL8	CONV8-512 MAXPOOL8	CONV8-384 MAXPOOL8	FS-CONV8-64	CONV8-128	FS-CONV8-256	CONV8-512
FC	FC	CONV8-384 MAXPOOL8	FS-CONV8-32	FC	FS-CONV8-128	FC
		CONV8-256 MAXPOOL8	FS-CONV8-5		FS-CONV8-5	
		FC				

CONV: convolution operation, MAXPOOL: Max pooling operation, FS-CONV: Fractionally-strided convolution, FC: Fully-connected

(b) Prediction Accuracy of the Models

Model A	Model B	Model C	Model D	Model E
76.2%	78.4%	81.3%	81.6%	83.2%

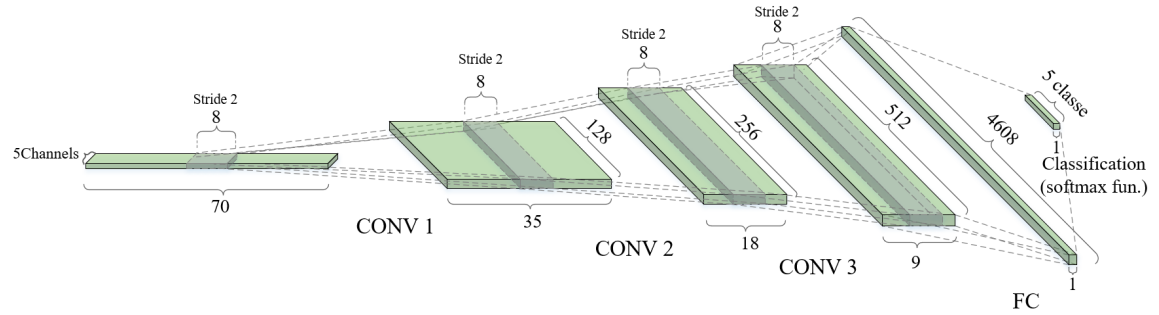
Table 5.1: Model Architectures and their Prediction Accuracy

Class-K, Fake].

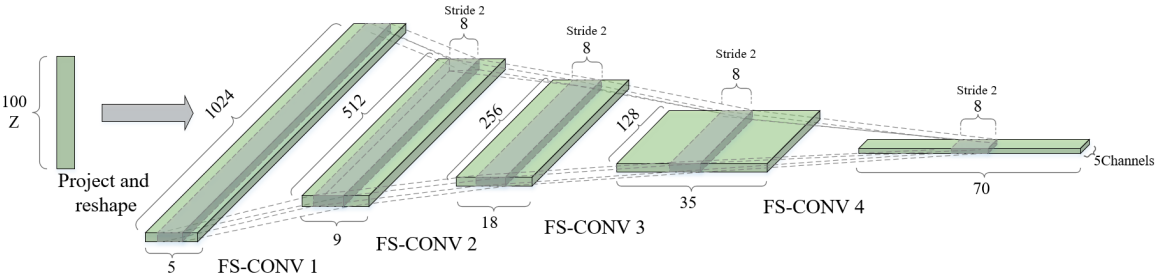
Model C is a CNN model based on the architecture developed by Krizhevsky [150] for Convolutional Neural Networks. We used the same number of kernels of CNN models corresponding to the values suggested by the aforementioned study.

The Leaky ReLU activation function [161] is applied to the output of every convolutional layer (the activation functions on hidden layers are not shown in Figure 5.3a and 5.3b. Also, we applied batch normalization [162] after each convolutional layer and dropout with keep probability of 0.5.

The generator architecture is shown in Figure 5.3b. In GANs modeling studies, usually the generator has the inverse architecture of the discriminator, i.e. generator starts from a noise and outputs fake samples that have the same size of the inputs to the first layer of discriminator. Hence, the best approach is to design the generator architecture in accordance with discriminator architecture, but in an inverse way. The generator begins with the input Z of size 100, which is sampled from a random uniform distribution. The input Z is reshaped and projected into a 1024×5 output. After, there are four fractionally-strided convolution layers. The output of the final layer is a 70-point segment with 5 channels which generates the fake samples. Finally, the generated fake samples, along with the 70-point segments from the MTL Trajet dataset, are fed to the discriminator as the input layer.



(a) Discriminator Architecture



(b) Generator Architecture

Figure 5.3: Architecture of Model E

5.4 Results

The results of models in Table 5.1a are shown in Table 5.1b. The CNN models A, B and C can achieve the prediction accuracy of 76.2%, 78.4% 81.3%, respectively. The semi-supervised DCGANs models D and E predict the travel mode with 81.6% and 83.2% accuracy. In total, the developed DCGAN models were able to predict the travel mode with higher accuracy than the CNN models.

One of the major issues related to generative models is evaluating the generated samples [99]. While in the image processing context the generated images can be evaluated visually, it is not the case for generated trajectory segments. Finding proper evaluation metrics for generated trajectories, with different architectures or hyperparemeters values, should be the focus of future studies. One of the disadvantages of the GANs models is their high computational time and memory requirements.

The supervised, unsupervised, and total loss for different numbers of training steps are shown in Figures 5.4-5.6. The supervised loss is the negative log probability of labels, given that the data come from the real dataset. The unsupervised loss is composed of two terms: a) the negative log of

1 minus the probability of fake labels given that the data is real, b) the negative log probability of fake labels given that the data is generated (being fake). The unsupervised loss function introduces an interaction between the generator and discriminator (i.e. the classifier), which is indeed the GANs game value [99]. As expected, the values of unsupervised loss are higher than those of the supervised loss, due to the fact that the generator, i.e. G(Z), produces samples by transforming the random noise vectors, i.e. vector Z in Figure 5.3b, to the real data distribution [99].

5.4.1 Comparison with previous studies

While comparing the prediction accuracy rates of different studies, several considerations should be taken into account. Such as differences in the quality of data across studies, the number of output categories (classes), and sample sizes. Moreover, the methods of validating the labels of observations in a dataset may affect data quality and the prediction accuracy rate of the developed models. The validation of the MTL Trajet data was carried out without any recall or surveyor-intervened validation process. Indeed, the respondents validated their travel mode through an in-app questionnaire. Although such a method of validation places less burden on respondents, it may reduce the quality of the labeled data. Furthermore, the MTL Trajet data comprises primarily GPS data (i.e. no accelerometer data) from user smartphones, to cut down on battery consumption.

Taking into account these considerations, our results are on par with previously obtained findings in the mode detection literature. Dabiri and Heaslip [39] have arrived at a test accuracy of 79.8% for their best Convolutional Neural Network model. Also, the ensemble of their best model was able to reach an accuracy as high as 84.8%. Zheng et al. [64] have developed a decision tree model with an overall accuracy of 76.2%. The aforementioned studies [39, 64] have used only GPS data from smartphones, similar to the data gathered by MTL Trajet. The results of our best semi-supervised DCGANs model outperforms the findings of both indicated studies, that is 83.2% versus 79.8% and 76.2% of prediction accuracy, respectively. Apart from the aforementioned differences between the databases and different modeling approaches among the studies, we think that DCGANs model improves the prediction accuracy of discriminator by generating more samples and enlarging the original MTL Trajet dataset.

Bantis and Haworth [118] built several classifiers to infer travel mode using GPS and accelerometer

data as well as user socio-demographics. Their hierarchical dynamic Bayesian network can reach an accuracy of 90%. However, their model was built on a training data containing trips from only 5 individuals. Apart from the different modeling approaches between our study and theirs, the richness of the accelerometer data together with the different sizes of datasets may give an explanation to differences in the prediction accuracy.

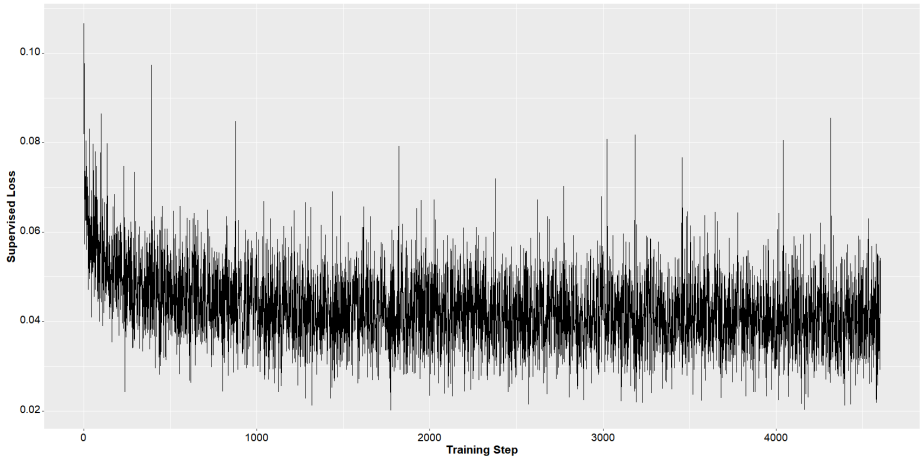


Figure 5.4: Supervised Loss of DCGANs Model for Different Numbers of Training Steps.

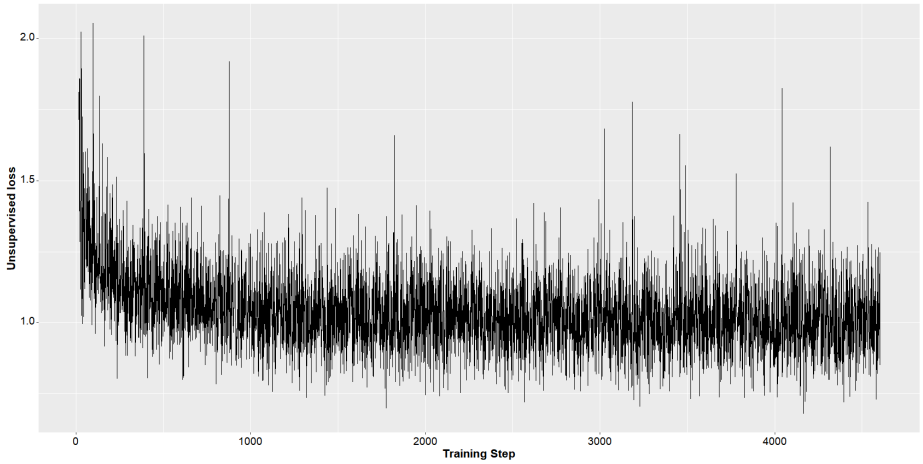


Figure 5.5: Unsupervised Loss of DCGANs Model for Different Numbers of Training Steps.

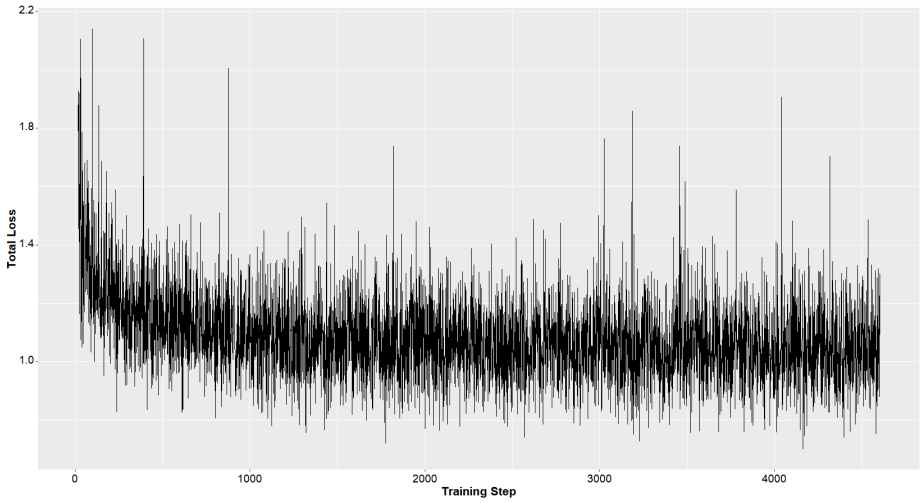


Figure 5.6: Total Loss of DCGANs Model for Different Numbers of Training Steps.

5.5 Conclusion and future work

Generative adversarial networks have proven their ability in unsupervised and semi-supervised image and text recognition. This study developed semi-supervised generative adversarial networks to infer transportation mode from GPS trajectories.

The developed semi-supervised DCGANs models share the same architectural innovations used in the image recognition literature. We have shown that similar architecture can be used in travel information inference from smartphone travel survey data. Generative models have the advantage of increasing the prediction accuracy of classifiers (as seen here) without increasing the number of labeled samples. The semi-supervised DCGANs model in this article shows a slightly higher classification accuracy than Convolutional Neural Network models.

In general, GAN models have gained a reputation for being difficult to train. We also investigated another alternative method for training GANs suggested by Chavdarova and Fleuret [163], referred to as SGAN, in which multiple pairs of adversarial networks, i.e. "local" pairs of generators and discriminators, are independently trained. Also, on top of them, there is a global discriminator and generator, which is trained with the corresponding "local" opponent in each epoch. We developed SGAN models with different number of local pairs, in which each local adversarial network was a DCGAN model, and a global generator and discriminator was trained against each of its local opponents. However, we found that the results of the SGAN in the context of mode of transport

inference were sufficiently unstable so as not to present them here.

In future work the framework developed in this study could be extended along the following dimensions. Apart from the semi-supervised DCGANs developed in this study, there are other types of semi-supervised GANs such as Conditional GANs [101], that enable the GANs generator to generate labeled samples, that could be tested. Examining different distributions to sample Z from while training the generator to see how it affects the performance of the discriminator. GAN requires more training time and memory resources compared to CNN or other machine learning models. The most time-consuming part of GANs is the gradient descent calculations applied simultaneously on both generator and discriminator. Developing methods that need fewer gradient descent steps could significantly decrease the computation time of GANs. The exploration of methods of inferring multiple modes for trips based on smartphone travel survey data. Creating an ensemble of DCGANs models to achieve higher prediction accuracies.

Mode prediction accuracies among studies have high degree of heterogeneity, due to different datasets and data quality, data preparation procedures, and classification approaches. While the prediction accuracies of the models in this paper are among the extremely high prediction accuracies in the literature, they are good relative to other comparable studies, in terms of dataset size and modeling approach, in the literature. As well, the fact that the dataset used was from a large-scale, real-word study likely introduces a great deal more variability than controlled, small-scale studies. Finally, the purpose of this paper was to demonstrate the use of DCGANS developed primarily for image processing in the context of mode inference with 1-D smartphone trajectory data. Future work will allow exploration of better performing models either with more channels and/or improved architectures.

Chapter 6

Multi-task Recurrent Neural Networks to Infer Mode and Purpose from Smartphone Trajectory Data

Preamble

This chapter introduces multi-task learning in the context of mode and purpose inference using Recurrent Neural Networks. As mentioned in the literature review, Section 2.1, at the time of writing this thesis we did not find any studies focusing on inferring mode and trip purpose simultaneously, in a unique modeling structure. Until now, the models developed in the domain of mode or purpose inference have focused on single-task learning approaches. This chapter takes advantage of sequential deep learners developed in Natural Language Processing (NLP), referred to as Recurrent Neural Networks (RNN). The RNN models in this chapter can be fed multi-input, such as disaggregate point-based features as well as socio-demographics and features related to the destination point (such as land-use data around the destination). Moreover, they produce multi-output, i.e. mode and purpose of trips. While the goal of this chapter is to demonstrate the potential of multi-task learning approaches in trip information inference, it also tests the performance of RNN models against the CNN or Random Forest models developed in Chapters 3 and 4.

Abstract

Multi-task learning is used to simultaneously infer mode of transport and trip purpose from traveler data collected as part of a smartphone-based travel survey. GPS trajectory data along with socio-demographics and destination-related characteristics are fed into a multi-input neural network framework to predict two outputs; mode and purpose. We deployed Recurrent Neural Networks (RNN) which are fed by sequential GPS trajectories. To process the socio-demographics and destination-related characteristics, another neural network, with different embedding and dense layers is used in parallel with RNN layers in a multi-input multi-output framework. The results are compared against the single-task learners that classify mode and purpose independently. We also investigate different RNN approaches such as Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU) and Bi-directional Gated Recurrent Units (Bi-GRU).

The best multi-task learner was a Bi-GRU model and able to classify mode and purpose with 84.33% and 78.28% of F1-measure, while the best single-task learner to infer the mode of transport was GRU model and achieved 86.50% of F1-measure, and the best single-task Bi-GRU purpose detection model reached 77.38% of F1-measure. The results demonstrate the multi-task learning approach slightly benefits to purpose classification, however, it did not improve the mode classification performance.

6.1 Introduction

During the last decade, transportation practitioners and planners have begun using the smartphone-based surveys to collect the respondent trajectories and other trip-related information. Researchers have investigated various inference methods to elicit the trip information from smartphone-based travel surveys, especially to detect the mode of transport and trip purpose.

In recent years, deep neural networks have demonstrated remarkable accomplishments in various fields of science, particularly in natural language processing and image recognition tasks. Concerning GPS trajectory analysis, deep learning approaches allow discovering factors related to each GPS point that are overlooked by traditional machine learning algorithms that use aggregated features for whole trips.

Deep learning models typically care about predicting a single task by training one model or an ensemble of models. However, by focusing on training a single-task model we may ignore advantageous representations derived from shared layers of related tasks. Taking into account the correlation between related tasks may improve the prediction accuracy of deep learning models. In the deep learning literature, this approach is called Multi-Task Learning [164]. Other names have been utilized in the literature in the guise of "Multi-task learning": learning with auxiliary information or tasks, joint learning, or learning to learn [164]. However, any deep learning model dealing with the optimization of two or more loss functions can be considered as a multi-task learner. Even supposing our goal is to optimize a single loss function, as is the case in many neural network implementations, we can benefit from auxiliary information or tasks to improve our main task [164, 165]. As Caruana [165] has summarized the goal of multi-task learning: "Multi-task learning improves generalization by leveraging domain-specific information contained in the training signals of related tasks".

In transportation demand analysis, trip purpose and mode of transport are arguably the most important trip characteristics. The importance of trip purpose can be drawn from a widely-accepted principal stating that "transportation is a derived demand" [25], which means the need for travel is derived from another need, for example the need for work, education, recreation etc. More explicitly, transportation demand is about satisfying a need for participating in an activity at the end of

the trip, i.e. the destination. Moreover, mode of transport and trip purpose are highly correlated [25]. For example, due to the limited number of parking spaces in downtown or Central Business Districts (CBD) in metropolitan areas, many travelers destined there use public transit for commuting. Hence, we can say work trips have high correlation with public transit, as mode of transport. Therefore, due to the correlation between mode and purpose, as well as the joint role of mode and purpose in stimulating the transportation demand, this study attempts to infer mode and purpose simultaneously from trajectory data using deep learning models.

We feed point-wise trajectory information as input into different multi-task Recurrent Neural Networks (RNN) architectures to infer mode of transport and trip purpose. As well, the models take advantage of single-observation auxiliary data, such as socio-demographics, to improve the prediction accuracy of mode and purpose of trips.

The study has been conducted using data collected by the smartphone travel survey app, MTL Trajet, which is an instance of the smartphone travel survey platform, DataMobile/Itinerum [8]. The MTL Trajet data was collected as part of a large-scale pilot study on the 17th of October 2016 in a 30-day travel survey study, in which over 8,000 respondents participated [18].

The rest of the paper is organized as follows: a background section describes previous work on multi-task learning and RNNs in mode or activity detection from trajectory data. The methodology section sets out the framework of the RNN models as well as of the data pre-processing. The next section after that presents the results of the different multi-task RNN architectures on the MTL Trajet dataset. The last section concludes the paper.

6.2 Background

This section reviews previous research related to mode and purpose detection from smartphone data and briefly introduces the RNN models, especially the Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). Comprehensive reviews on mode and purpose detection have been conducted by Wu et al. [121], Elhoushi et al. [147], and Wang et al. [15]. Different Recurrent Neural Network approaches have been explained in detail by Goodfellow et al. [33]. Also, recent applications of multi-task learning have been fully covered by Ruder et al. [164].

Mode and purpose detection has been studied in the literature using various approaches, which include: rule-based, machine learning, deep learning, and discrete choice approaches. Ensemble tree-based methods [40], Random Forest models [153], hybrid rule-based and Random Forest approaches [56] and other machine learning models have been widely used in the literature to detect mode of transport. Methods from other fields of science have been applied for the mode detection task. For example, Assemi et al. [57] utilized a nested logit model to infer mode of transport from smartphone travel surveys. The next section reviews the studies on mode and purpose detection from mobile phone data using deep learning approaches.

6.2.1 Mode and Purpose Detection

In recent years, deep neural network models, such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks have been deployed for mode detection. Convolutional Neural Networks (CNN) used by Dabiri and Heaslip [39] and Yazdizadeh et al [166] to detect the mode of transport. Endo et al. [58] suggested a framework to automatically calculate trajectory features by converting GPS-points along a trip into a two-dimension (2D) image structure. They have then deployed a deep learning model and fed it with the "stay time" of each GPS point (as a correspondence to pixel values of an image). Finally, they used traditional machine learning models, such as logistic regression and decision trees, to predict mode of transport based on pixel values and other hand-crafted features.

Recurrent Neural Networks, Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU), control gate-based Recurrent Neural Networks (CGRNN) have been applied on smartphone data to detect the mode of transport [167, 168]. Vu et al. [167] analyzed different RNN approaches and found the superiority of GRU and CGRNN models over the simple RNN and LSTM models to infer the mode of transport from accelerometer data. Simoncini et al. [169] utilized the RNN model for vehicle classification from low-frequency GPS data[1]. They used different features, such as timestamp, longitude and latitude, speed and odometer to detect vehicle type. They have used an RNN architecture with a 1-D pooling layer, which aggregates the final recurrent layer of RNN, to

[1] Vehicle classification studies focus on determining the type of vehicle, for example categorizing vehicles based on the number of axles.

predict vehicle type.

Liu et al. [170] developed a bidirectional LSTM (Bi-LSTM) to classify mode of transport from GPS data. The Bi-LSTM model demonstrated good performance while only considering time interval, longitude, and latitude of trajectory GPS data. Moreover, they suggested an enhanced embedding Bi-LSTM model that considers the time interval as an external feature fed into an embedding layer, where the Bi-LSTM part uses the same architecture as their base Bi-LSTM model.

With respect to purpose detection, there are fewer studies in the literature compared to mode inference studies. While mode detection is related to GPS-point features, for example, speed or acceleration, trip purpose has more correlation with trip destination attributes, such as different land-use types in the vicinity of a destination, socio-demographics, and other trip attributes such as time-of-day or day-of-week of the [153]. While trip trajectories can be treated as sequential observations and fed to RNN or CNN models to detect the mode of transport, the trip destination is a single-point observation, and cannot be fed to RNN or CNN models. This limitation has caused researchers to use more classical machine learning algorithms, such as Random Forest [107, 153], or rule-based methods [80] for purpose detection. However, due to the correlation between mode and purpose in travel choice behavior, one may expect better prediction accuracies while training a single multi-task model, rather than training two single-task learners. Multi-task learning has been used in traffic data imputation studies [83]. Rodrigues et al. [83] proposed multi-output Gaussian processes (GPs) to model the spatial and temporal patterns in traffic data. They demonstrated that the multi-output GP model is able to capture the complex dependencies and correlations between nearby road segments to improve imputation accuracy. Their multi-output model outperforms other imputation methods, such as a Recurrent Neural Network model, by taking into account the complex spatial dependencies between subsequent road segments.

This study investigates using a multi-task approach, i.e. a multi-task learner, based on both trip trajectory and destination characteristics, and socio-demographic information. By implementing a multi-task learning approach, We aim to capture and exploit the correlation between mode of transport and trip purpose in our dataset and find out to what extent such an approach can enhance the prediction performance for both mode and purpose of trips. The model architecture is explained in Section 6.3.

6.2.2 Recurrent Models

Recurrent Neural Networks have been a popular choice for training sequential data [33]. However, their capability is restricted by a lack of memorization of long and short term dependencies across sequential data points, due to the vanishing or exploding of gradients in long sequences [171]. LSTM and GRU approaches were developed to overcome these shortcomings of RNN models. We briefly explain each component of LSTM and GRU models and their differences.

Inside an RNN Block

In the core of each RNN model, there is a recurrent hidden state described by [172]:

$$h_t = g(Wx_t + Uh_{t-1} + b) \tag{19}$$

where x_t is the input vector (in our case a vector of GPS points along a trip) at time t, h_t is the hidden state at time t, g is any activation function, such as tangent hyperbolic, W and U are the trainable weights, and b is the trainable bias. Finally, h_{t-1} is the hidden state (or output) of the previous time step $t-1$. Hence, each RNN block has two inputs: x_t and h_{t-1}, and outputs the hidden state at the current time step, i.e. h_t to the next RNN block.

Inside an LSTM Block

The LSTM model introduces three gating (control) signals: *input*, *forget* and *output* gating signals at each time step t. These gating signals are similar to the Equation 19, with their weights and bias parameters, and a sigmoid activation function. One can imagine these gating signals as valves that control the flow of water in pipelines. However, these valves, with sigmoid functions, control the flow of memory, input, and output between LSTM blocks.

The *forget, input* and *output* gating signals have the following equations:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{20}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{21}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{22}$$

where σ is sigmoid function.

Moreover, LSTMs benefit from using internal memory cell, $\widetilde{c}_t$, inside each LSTM block. The flow of memory between blocks is determined by *forget* and *input* control signals. Indeed, an LSTM block can forget the memory, when the output of the sigmoid function of a *forget* gating signal is 0, or keep the memory if the sigmoid outputs 1.

As shown in Equation 20, the *forget* gating signal f_t is calculated based on the hidden state of previous time step h_{t-1}, and the input of the current LSTM block x_t. The output of the sigmoid function of the *forget* gating signal f_t is then applied on the old memory cell c_{t-1} by a element-wise multiplication, as in Equation 23:

$$c_t = f_t \textcircled{e} c_{t-1} + i_t \textcircled{e} \widetilde{c}_t \tag{23}$$

The element-wise multiplication is denoted by $\textcircled{e}$. The second term in the above equation controls how much of the memory in current memory cell $\widetilde{c}_t$ should influence the new memory, i.e. c_t. Indeed, this role is played by the *input* gating signal while multiplied (element-wise) with the $\widetilde{c}_t$. The current memory cell $\widetilde{c}_t$ is defined as:

$$\widetilde{c}_t = g(W_c x_t + U_c h_{t-1} + b_c) \tag{24}$$

where g is usually the hyperbolic tangent function or the rectified Linear Unit (ReLU) [172].

After calculating the current memory cell of an LSTM block by Equations 20, 21, 23 and 24, the hidden state, h_t of the LSTM block is obtained by the following formula:

$$h_t = o_t \textcircled{e} g(c_t) \tag{25}$$

where the o_t is the *output* gating signal, defined by Equation 22, which controls how much memory should transfer to the next LSTM block.

As explained in the above equations, the distinctive characteristic of the LSTM model is the concept of the memory cell, which is passed through LSTM blocks and the flow of memory is regulated by the implementation of different gating signals.

Inside a GRU Block

The main difference between an LSTM and GRU block is the number of gating signals, with two gating signals in GRU instead of three gating signals in LSTM. GRUs possess an update gate, denoted by z_t, and a reset gate, r_t [172]. The hidden state of a GRU block is calculated based on the following equations [172]:

$$h_t = (1 - z_t) \circledcirc h_{t-1} + z_t \circledcirc \widetilde{h}_t \tag{26}$$

$$\widetilde{h}_t = g(W_h x_t + U_h(r_t \circledcirc h_{t-1} + b_h) \tag{27}$$

The two gating signals in GRU blocks are defined by similar formulas for the gating signals in an LSTM:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{28}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{29}$$

Various studies have demonstrated the comparable performance of GRU and LSTM in many cases [173]. Moreover, some studies have shown GRU outperforms the LSTM models [172, 173].

6.3 Methodology

In this section, we describe the data used, data pre-processing, the RNN architectures used and hyperparameter value determination.

6.3.1 Data

The MTL Trajet dataset consists of three types of data; validated mode and purpose of trips, GPS coordinates, *timestamps*, and respondent socio-demographic information. The MTL Trajet app is supported by an in-app mode and purpose validation process, which prompted and asked respondents to validate their mode of transport and trip purpose upon detecting a stop during their movements (stops are detected by a rule-based algorithm that works in the background of MTL Trajet app). Furthermore, upon the installation of MTL Trajet, a series of questions are asked of respondents to gather their socio-demographics, such as age, sex, occupation, home location, work or school location, etc. During the MTL Trajet 2016 survey, over 33 million GPS points were recorded. There were four modes of transport, i.e. walk, bike, car, public transit, validated by respondents. Also, six trip purposes were collected including education, health, return home, shopping, work and return home.

To detect the trips, we utilized a rule-based trip-breaking algorithm explained in Patterson & Fitzsimmons [109]. The algorithm detects start and end point of trips based on the 3-minute dwell time between GPS points as the most prominent criterion for detecting trips. Afterward, the algorithm verifies the velocity and parameters relating to the public transit network (i.e. transit junctions and metro station location) and then stitches segments back together into complete trips. As an example, wherever two consecutive GPS points are located within 300 meters of two different metro stations (due to the sparsity of data collection when underground), and the time interval between them is less than the maximum travel time by metro, the segments are considered as part of the same trip. Also, wherever two consecutive GPS points fall within the same intersection with bus correspondences, the algorithm allows for a wider time interval (10-minute instead of 3-minute gap).

6.3.2 Data Preparation

The dataset consists of two type of data: sequential data (trip trajectories), that comes from the GPS points, and auxiliary data, i.e. single-observation data including socio-demographics and trip destination characteristics. Each trip in our dataset comprises GPS points with three GPS features: time interval, longitude, and latitude, used in the sequence part of our models. While the number

of GPS points along a trip varies between 15 to more than 1000 points, we considered only the 70 points of each trip (as 70 is the average number of points along all trips in our dataset). Also, to select the 70 points for each trip, we considered the first 35 and the last 35 GPS points along each trip. When the number of points along a trip was less than 70, we padded the trajectories.

While in Chapter 3, we analysed mode and purpose separately, in this chapter, the focus is on trips for which both mode and purpose have been validated. As some of the trips have not been validated for both mode and purpose, our data size decreased to 7,763 trips. The auxiliary features used are shown in Table 6.1. Socio-demographics and trip characteristics are obtained from the MTL Trajet dataset. Land-use data are derived from Montreal land-use data (compiled by the provincial government ministry "MAMROT" [133]). Foursquare data consists of *checkinsCount* and *usersCount*, where *checkinsCount* is the total number of check-ins in a venue near a trip destination, while *usersCount* accounts for the total number of users who have ever checked into a venue near a trip destination (in a 250-meter vicinity).

6.3.3 Time Transformation

Regarding GPS features, we treat the recorded time of each GPS point as the number of seconds after midnight, which is inherently a cyclical feature. For example, 5 minutes after and before midnight, are recorded as 300 and 86,100 seconds, respectively. While the time interval between these two times is only 600 seconds, their values in seconds suggest a too-large time interval, i.e. 85,800 seconds, which does not represent the cyclical behavior of time during a day. To let the neural network recognize that an attribute is cyclical, one can transform the time into two dimensions, using the cosine and sine functions, as follows:

$$sinusoid\_time = sin(2 * \pi * seconds/total\_seconds) \tag{30}$$

$$cosinusoidal\_time = cos(2 * \pi * seconds/total\_seconds) \tag{31}$$

where the *seconds* is the number of seconds after midnight, and *total_seconds* is the total number of seconds of each day. For example, 5 minutes after midnight will represent as 0.021 for the sinusoid dimension, and 0.999 for the cosinusoidal dimension. Also, 5 minutes before midnight will be

Table 6.1: Auxiliary Features Used in Mode and Purpose Detection.

Attribute	Definition	Embedding size
Trip Characteristics		
DAY of WEEK	1-7 for Monday through Sunday	4
HOUR_START	Trip start hour, ranging from 0 to 24	12
HOUR_End	Trip end hour, ranging from 0 to 24	12
CBD_ORIGIN	1: if the origin is located in Montreal's CBD, 0: otherwise	1
CBD_DESTIN	1: if the destination is located in Montreal's CBD, 0: otherwise	1
HOME_DEST	Direct distance between trip destination and individual home location	-
STUDY_DEST	Direct distance between trip destination and individual education location	-
WORK_DEST	Direct distance between trip destination and individual work location	-
HOME_ORG	Direct distance between trip origin and individual home location	-
STUDY_ORG	Direct distance between trip origin and individual education location	-
WORK_ORG	Direct distance between trip origin and individual education location	-
MTL_ORIGIN	1: if the origin is located in Montreal Island, 0: otherwise	1
MTL_DESTIN	1: if the destination is located in Montreal Island, 0: otherwise	1
Trip Destination Features		
Land-use (number of land-use parcels in 250 meters around a trip destination)		
LU_*	23 different attributes each one shows the frequency of the corresponding land-use category	-
Foursquare *checkinCounts* (number of checkinCounts in 250 meters around a trip destination)		
CH_*	10 different attributes each one shows the checkinCounts for the corresponding Foursquare category	-
Foursquare *userCounts* (number of usersCounts in 250 meters around a trip destination)		
UC_*	10 different attributes each one shows the usersCounts for the corresponding Foursquare category	-
Socio-demographics		
AVG_PRICE_NEIGH	The average value of residential buildings around each individual's home	-
SEX	0:male, 1: female, 2: other/neither	2
OCCUPATION	0:full-time worker, 1: part-time worker, 2: Student, 3: Student and worker, 4: Retired 5: At home	2
AGE	0: age between 16-24 ,1: 25-34 , 2: 35-44, 3: 45-54, 4: 55-65, 5: 65+	3

transformed into -0.021 for the sinusoid dimension, and 0.999 for the cosinusoidal dimension. Such transformation enables the neural network to consider the cyclical character of time over 24 hours.

6.3.4 Entity Embedding

We utilized the entity embedding approach [174] for the categorical data related to the trip destination or socio-demographics, instead of the one-hot Encoding approach. Both approaches have been utilized by researchers to deal with categorical data where there is no intrinsic ordering to the categories. The advantage of using entity embedding over the one-hot encoding is that each categorical variable is mapped into a higher dimension space which is richer in capturing the correlation

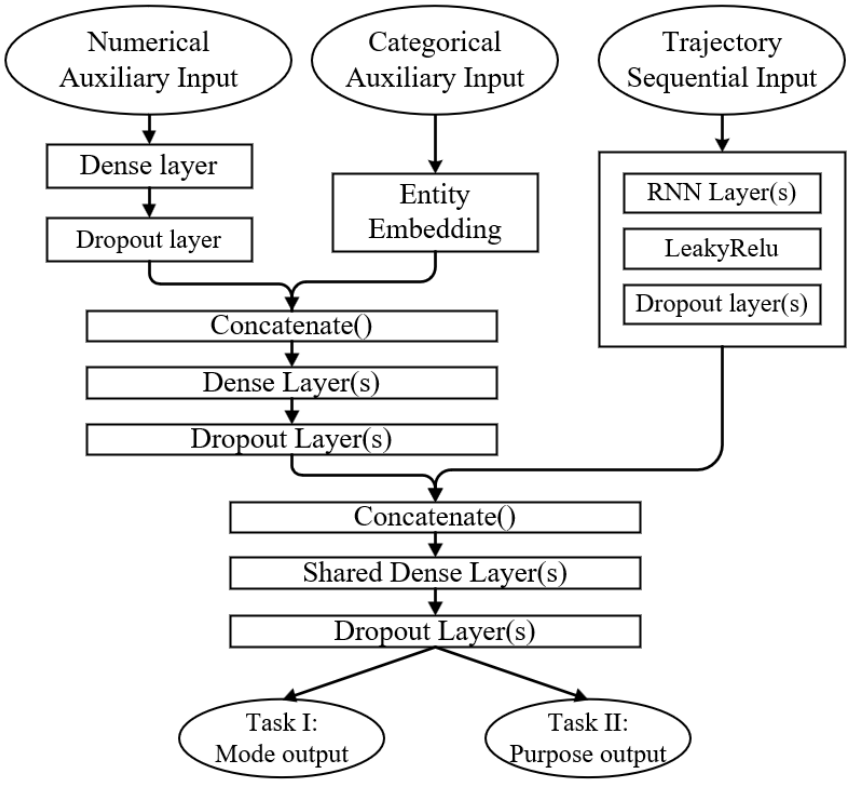


Figure 6.1: Schematic Framework of a Multi-input Multi-output Learner to Infer Mode and Trip Purpose.

between different levels of a variable. For example, “days of the week” is usually coded as a categorical variable in a 1 to 7 scale. However, with entity embedding each day of the week is usually mapped into 4-dimensional vector space enabling the model to capture the similarities, for example between “Saturday” and ”Sunday”.

The dataset in the current study contains 17 categorical and 50 numerical non-sequential attributes. The categorical attributes are first fed into an embedding layer and the output is then concatenated with the output of non-sequential numerical data, as shown in Figure 6.1.

6.3.5 Model Architecture

This section explains the architecture of the models to examine the performance of multi-task and single-task learners. As mentioned in the previous sections, the dataset in the current study consists of two types of data: the sequential data, which is the sequence of GPS points along a trip trajectory,

Table 6.2: Hyperparameter values used in the Study

Name	Hyperparameter value
Recurrent layers	Types:LSTM, GRU, Bi-LSTM, Bi-GRU, Num. of layers: 1, 3, 6 Num. of nodes: 70, 210, Activation function: Leaky ReLU
Activity Funciton	Relu, LeakyRelu($\alpha = 0.05$)
Dropout	0.2, 0.25, 0.5
Output layer	Activation: Softmax for both mode and purpose of trip
Optimization method	Adam optimizer
Batch size	16, 32, 64, 128

and the single-observation data, which are the characteristics of the trip destination and the socio-demographics of travelers. To deal with the sequential data, we investigated different Recurrent Neural Networks architectures, developed by researchers across various fields. Among them, Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) have gained the lion's share of attention. Also, we investigate the bi-directional GRU (Bi-GRU) to probe how the bi-directional architecture may improve the performance of the model. We also tested different hyperparameter values in Table 6.2.

Our approach toward the single-observation data, which contains two data types: categorical and numerical, is to develop entity embedding layers to process the categorical data, and dense layers to analyze the numerical data. A schematic of developed multi-input architecture to process all data types, i.e. trajectory sequential data, categorical auxiliary data, and numerical auxiliary data have been shown in Figure 6.1. The output of entity embedding layer and dense layers is concatenated and passed through further dense and dropout layers. Finally, the outputs from trajectory and auxiliary data are concatenated and fed into shared dense layers.

In the multi-task learning framework, the output of the shared dense layers in Figure 6.1 is fed into two cross-entropy loss functions, for generating labels for mode and trip purpose. In single-task models, the output of the shared dense layers is fed into one loss function, for either mode or purpose classification.

6.4 Results

This section presents the results of different single- and multi-task learners. We developed three series of models: single-task purpose classifiers, single-task mode classifiers, and multi-task mode and purpose classifiers. For each type of these classifiers, three different RNN architectures (i.e. LSTM, GRU, and Bi-GRU), for the RNN part of the framework have been tested. The accuracy, precision, recall and F1-measure of the developed models have been shown in Table 6.3.

To compare the single- and multi-task learners on a fair ground, the characteristics of the developed models were kept similar among all. The RNN part of models consists of three 70-node recurrent layers, each of which followed by a dropout layer. The dense layers for analyzing the auxiliary numeric data in Figure 6.1 compromise of one 256-node dense layers. Afterwards, the output of the final dense layer of numerical auxiliary input, is concatenated with the output of the entity embedding layer, which process the categorical auxiliary input. Subsequently, three 128-node dense layer process the output of concatenate layer. Afterwards, the results of RNN layers is concatenated with the results of the numerical and categorical input data. Later, the output of three 64-node dense layers, each of which followed by a dropout layer, amount to the shared dense layers of the framework in Figure 6.1. We tested different dropout probability values, i.e. 0.2, 0.25 and 0.5, and find the dropout layer with a probability of 0.2 results in better performance after RNN layers. However, after dense and shared layers, dropout layers with a probability of 0.5 demonstrated higher performance. While these characteristics are hold among multi-task and single-task learners, the output of the final layer of multi-task learners is fed into two cross-entropy loss functions. The two loss functions are then summed up, with equal weights.

The results of single-task purpose and mode classifiers as well as multi-task models are presented in Sections 6.4.1- 6.4.3. All the models are trained to epoch 200, and the results have been presented in Figure 6.2. Obviously, the performance of almost all the models in Figure 6.2 does not improve beyond the 100 epochs. Hence, we reported and compared the performance of the models on the test data for epoch 100 in Table 6.3, in order to compare them on a fair ground.

All the models are implemented in Python, using the Keras backend API with GPU support. Different hyperparameter values were tested to select the best model architectures and configurations.

Table 6.3: Prediction Results of Different Purpose and Mode Classifiers (Epoch = 100, test data)

Single/Multi Task	Task	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Measure (%)
Single-task Learners	Mode classification	LSTM	84.65	86.58	83.75	85.11
		GRU	86.07	87.49	85.58	86.50
		Bi-GRU	86.07	86.82	85.37	86.07
	Purpose Classification	LSTM	75.48	82.79	70.57	75.98
		GRU	75.99	83.62	71.02	76.59
		Bi-GRU	77.08	81.41	73.93	77.38
Multi-task Learners	Mode classification	LSTM	79.91	83.50	76.67	79.83
		GRU	84.08	85.94	81.52	83.59
		Bi-GRU	84.08	86.24	82.61	84.33
	Purpose Classification	LSTM	76.70	84.67	69.79	76.20
		GRU	77.34	85.56	71.44	77.61
		Bi-GRU	77.92	84.41	73.34	78.28

Models are trained on the Google Colab supported GPUs with 12.0 GB of RAM.

6.4.1 Single-task Mode of Transport Classification

We developed three baseline single-task mode inference models to compare the performance of multi-task learners in comparison with them. The accuracy, precision, recall and F1-measure of single-task purpose classifiers have been demonstrated in Table 6.3. With respect to F1-measure, the GRU model demonstrates highest performance, with F1-measure equal to 86.50%, comparing to LSTM and Bi-GRU models. Regarding the accuracy, the GRU and Bi-GRU demonstrated equal accuracy, 86.07%, and both superior to the single-task LSTM mode classifier. Also, GRU and Bi-GRU models achieved better recall values, 85.58% and 85.37%, compared to LSTM recall which is equal to 83.75%. Based on the results of single-task mode classifiers in Table 6.3, the GRU models demonstrated a better performance comparing with LSTM and Bi-GRU.

6.4.2 Single-task Trip Purpose Classification

The LSTM, GRU and Bi-GRU models are developed testing their performance on single task trip purpose classification. Table 6.3 demonstrates accuracy, precision, recall and F1-measure of single-task purpose classifiers. The Bi-GRU model achieved the highest F1-measure of 77.38%, while LSTM and GRU models demonstrated F1-measure of 76.59% and 75.98%, respectively. Moreover, the Bi-GRU model is superior to GRU and LSTM with respect to recall.

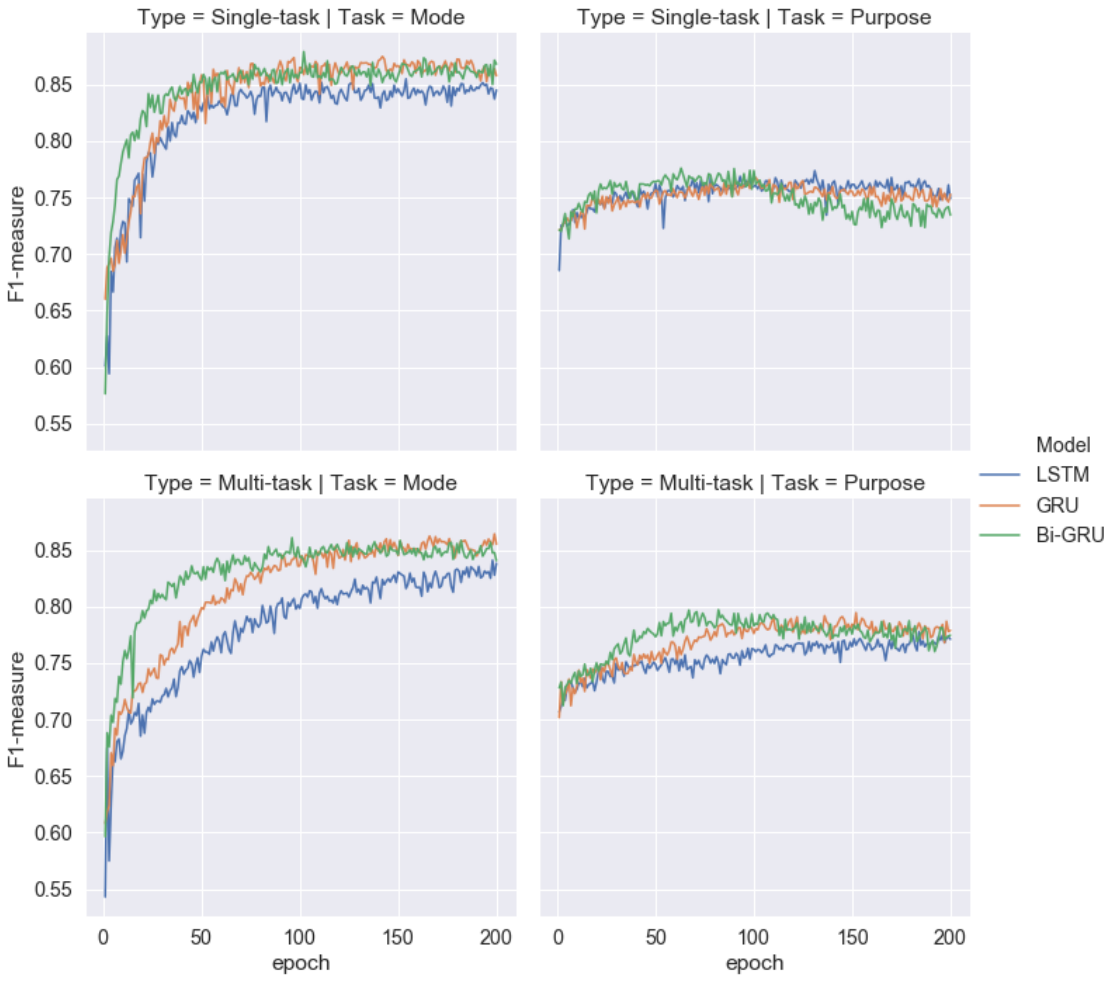


Figure 6.2: F-1 Measure of Different Classifiers over Epochs (top row left: single-task mode, top row right: single-task purpose, bottom row left: multi-task mode, and bottom row right: multi-task purpose classifier)

6.4.3 Multi-task Mode and Purpose Classification

Multi-task learners developed in this thesis posses the same architecture of single-task learners, except for the final loss functions, where two cross-entropy loss functions generate the final class labels. Optimization of the whole network is carried out by deploying an Adam optimizer on the sum of cross-entropy loss functions for mode and purpose.

As shown in Table 6.3, the F1-measure for multi-task LSTM learner to classify the mode of transport and trip purpose is %79.83 and %76.20, respectively. Comparing with F1-measure of single-task

LSTM mode classifier, which was 85.11%, shows a considerable drop in the classification performance of the LSTM model when being deployed in a multi-task framework. However, the multi-task LSTM model predicted the purpose of trips with a F1-measure a bit higher than the single-task LSTM purpose classification.

With respect to the multi-task GRU classifier, the model classifies the mode and purpose with F1-measure of 83.59% and 77.61%, respectively. Comparing these results with the single-task mode classifier shows that mode classification does not benefit from a multi-task framework, as single-task GRU achieved a F1-measure of 86.50%. Nonetheless, similar to the LSTM case, the multi-task GRU classifier depicts higher F1-measure comparing to single-task GRU when classifying trip purpose.

Regarding the multi-task Bi-GRU classifier, the model can achieve the F1-measure of 84.33%, which is higher than the F1-measure of both multi-task LSTM and GRU mode classifiers. However, mode classification with the single-task Bi-GRU classifier shows superiority over the multi-task Bi-GRU classifier. Similar to multi-task LSTM and GRU purpose classifiers, the multi-task Bi-GRU purpose classifier demonstrates a slightly better performance compared to the single-task Bi-GRU model.

Based on the results in Table 6.3, the highest F1-measure for mode classification belongs to the single-task GRU model, while among all the purpose classification models the multi-task Bi-GRU achieved the highest F1-measure.

6.4.4 Comparison with other Learning algorithms

This section compares the performance of algorithms developed in the previous sections against other machine learning approaches: Random Forest (RF) and Convolution Neural Networks. Two RF models are developed to infer mode of transport and trip purpose. As the point-based GPS information cannot be fed to the RF models, the aggregated features of the whole trip have been used to predict mode of transport. The socio-demographics also have been used to enhance the prediction accuracy of the RF model. For trip purpose inference with the RF model, a single-task RF model is developed using land-use, Foursquare and socio-demographic data. The best RF model to predict the mode of transport achieves F1-measure of 85.68%, while the RF model for trip purpose

prediction reaches F1-measure of 71.53%. The single-task GRU and Bi-GRU mode classifiers can predict the mode of transport better than RF model, 86.5% and 86.07%, respectively. However, the RF model shows better classification performance compared to all other mode classifies in Table 6.3. Nevertheless, comparing the RF purpose classier with the the models in Table 6.3 reveals the superiority of all single- and multi-task purpose classifier over RF model. We also developed a Convolutional Neural Network (CNN) model, with the same characteristics as the best single model explained in Chapter 4, to detect mode of transport. The model can achieve F1-measure of 79.13% to detect mode of transport.

6.5 Conclusion

Multi-task learning is known as a powerful inference method in machine learning [175]. Specifically, where there is a considerable correlation between multiple tasks, predicting them in a unique framework can potentially enhance prediction results. This research developed several single-task models to compare their results against multi-task learners to infer the mode of transport and purpose of trip from smartphone-based travel survey data. Based on the literature, mode of transport has been largely explained by point-wise GPS features, such as time, longitude and latitude. we also deployed features related to trip destination and socio-demographics to infer mode and trip purpose. We considered the LSTM, GRU and Bi-GRU modeling approachs, which are well-known RNN methods in the Natural Language Processing literature. The results of single-task mode detection models showed that the GRU approach can slightly better predict mode over the LSTM and Bi-GRU model. Surprisingly, the bi-directional approach, i.e. the Bi-GRU, implemented on both LSTM and GRU, does not improve the performance of the single-task mode classifier. We tested several architectures for the RNN models, and found that models with three RNN layers and three dense layer at the end, produced the best results. To prevent over-fitting, we applied a drop-out of p=0.20 at the end of each RNN layer, and a drop-out of p=0.50 at the end of each dense layer. The best model to detect the mode of transports is the single-task GRU model, which can detect the mode of transport with 86.50% of F1-measure.

The single-task purpose classifiers were deployed using the same modeling architectures as mode

classifiers. The best single-task purpose classifier was a Bi-GRU model with 77.38% of F1-measure. Unlike mode of transport classification, trip purpose classification benefits when predicted simultaneously in the same framework with mode of transport. The multi-task Bi-GRU model achieved the F1-measure of 78.28% to classify trip purpose. However, mode of transport classification in a multi-task framework did not gain the same advantage.

This study also examined the performance of the well-known RNN architectures, i.e. LSTM, GRU, and Bi-GRU in the field of transportation data inference. The results demonstrated that for the single-task mode classifier, the GRU slightly outperforms the LSTM and Bi-GRU architecture. However, almost for all the other models, i.e. single-task purpose classifier and multi-task mode and purpose classifier the Bi-GRU model showed higher performance over LSTM and GRU.

Furthermore, this research used an entity embedding approach to encode categorical data, such as time of day and day of the week, to capture the correlation between different levels of each category. All the embedded layers were trained simultaneously with sequential GPS trajectory data and other numeric non-sequential data. However, while land-use and social network data around each trip destination are included in the models, the study lacks the inclusion of such information around GPS trajectories. That is, for example, infrastructure data, such as whether a metro line or highway or bike route is in the vicinity of a GPS trajectories, which may help the algorithm to detect mode of transport more accurately. Finding methods to include all such information around each GPS trajectories is left to future studies.

Moreover, besides the target tasks, such as mode and purpose in the current study, some studies [175] have proposed including auxiliary tasks in modeling procedures. Auxiliary tasks are those not as important as the target task for the researcher, but they may improve the performance of the inference model on target tasks. In the current study, the socio-demographics that we asked from travelers, such as “occupation”, could play the role of auxiliary tasks instead of being fed as an input feature to the models. Examining such approaches may also help the future studies to develop inference models, that achieve higher-performance.

Chapter 7

Conclusion

For several decades the Household Travel Survey (HTS) along with transportation demand modeling have been used as central tools in transportation planning. With the rapid changes in transportation technologies, transportation practitioners require the collection of data more frequently over larger metropolitan regions. Traditional HTS methods are costly due to human resource requirements and suffer from several other disadvantages. Moreover, they are not well adapted for cities aiming to collect travel data more frequently, for example on yearly or mid-yearly basis. Over the last two decades, new technologies such as GPS systems, now embedded in smartphones, have begun to provide transportation planners with new data collection tools. Such mobile phone technologies have attracted more attention over the last decade due to the pervasiveness of mobile phones and higher quality sensors and systems embedded inside them.

However, while the smartphone-based travel surveys potentially offer great benefits over traditional HTS methods, inferring trip information from collected data has proven to be more difficult than at first hoped for. Many studies have examined the application of artificial intelligence in inferring trip information from trajectory data. This thesis has addressed a few limitations in the literature and has considered the future directions and challenges for inference methods applicable to trajectory data.

With this aim, the study first identified the trip information to be inferred from trajectory data. The "mode of transport", "trip purpose" and "transit itinerary", are among the most important types of information for use in transportation demand modeling and transportation network design, as

explained in Section 1.2.

Afterwards, we identified the following limitations of the current literature, namely:

- The potential of smartphone-based travel surveys as a large-scale real-world travel survey have not been studied enough, due to the lack of large-scale datasets.
- The majority of studies have focused on using aggregate trip-based approaches
- Few studies have addressed the application of semi-supervised methods
- Transit itinerary inference has not been covered enough
- Multi-task learning for mode and trip purpose detection have not been applied on trajectory data
- Sequential learner algorithms have been understudied in trip information inference.

Based on these limitations, the contributions of this thesis are explained in the next section. The contributions section is then followed by sections describing the limitations of the research and future work directions.

7.1 Contributions

We used the MTL Trajet dataset to demonstrate the potential in smartphone-based applications for conducting real-world, large-scale travel surveys. The Random Forest models developed in Chapter 3 showed that a smartphone-based travel survey platform, such as Itinerum, if backed by a series of AI algorithms, can collect and produce the most important trip information with acceptable prediction accuracies (although there is room to improve the performance of the algorithms, especially for purpose detection). The mode detection Random Forest model was found to predict mode of transport with 87% accuracy. With respect to trip purpose, the Random Forest model attained an accuracy of 71% by taking advantage of complementary data sources such as land-use data, social media data (Foursquare) and General Transit Feed Specification (GTFS) data. Indeed, inferring trip purpose only based on trip features does not demonstrate a strong prediction accuracy and the above mentioned complementary data and socio-demographics are essential in achieving higher prediction

rates for trip purpose detection. Also, a Random Forest model developed to infer transit itinerary achieved an 80% prediction accuracy. Transit itinerary has been covered very little in the literature and requires more attention. Indeed, the RF model in this thesis is the only AI approach, at the time of writing this thesis, applied for transit itinerary inference.

Afterwards, Convolutional Neural Networks (CNN) were developed to infer mode of transport by using disaggregate point-based features, the GPS points along a trajectory. Considering each GPS point along a trajectory in a similar way to the way pixels are analyzed in image processing with CNN models was our first attempt to consider point-based features to predict mode of transport. Moreover, the application of ensemble methods was examined to enhance the performance of CNN learners. Indeed, considering GPS points along a trajectory as pixels in an image opened the door to benefiting from the many other algorithms developed in the field of image recognition and more attention and research is required to reveal how much the travel data field can benefit from algorithms developed in the domain of image recognition.

The Random Forest and CNN models in Chapter 3 and 4 have been deployed as supervised learners. Semi-supervised learning was considered by applying semi-supervised Generative Adversarial Networks to detect mode of transport. Based on our experience with CNN architecture as a supervised learner in Chapter 4, we used the convolutional architecture as the discriminator in the GAN framework. However, the semi-supervised GAN was not found to achieve remarkable improvements of performance over the supervised approach. GANS are well-known for being hard to train, and it seems that more effort is required to take advantage of GAN architecture for information inference from smartphone survey data.

Besides considering GPS points as analogous to pixels, we considered the sequence of GPS points along a trip as analogous to the sequence of words in a sentence. Such similarity enabled us to deploy the sequential learners developed in the realm of Natural Language Processing (NLP) to infer the mode of transport. We developed two of the most well-known RNN models, i.e. GRU and LSTM models, to infer mode of transport, and found that GRU slightly outperforms LSTM when applied to mode detection.

Trip purpose was also inferred from smartphone data using neural network models (with embedding layers for analyzing the categorical data, such as socio-demographics, and trip-related data). These

models were found to be better able to infer trip purpose than the RF model.

To examine the use of multi-task learning in purpose and mode inference, a multi-input, multi-output framework was developed using GRU and neural network layers. The multi-task learning framework was fed by two data types: (a) GPS trajectory data fed into the GRU layers, and (b) auxiliary data, i.e. categorical and numerical socio-demographics and destination related data, fed into embedding and neural network layers. Moreover, the multi-task learner outputs two tasks: (a) mode of transport and (b) trip purpose. The multi-task GRU learner resulted in slightly better prediction performance for trip purpose, over single-task GRU learner. However, the performance of multi-task GRU classifier to infer mode of transport was not as good as the single task GRU classifiers. We also compared the results of the single-task GRU classifier with the single-task CNN and RF models developed in this thesis to infer mode of transport. The single-task GRU classifier outperforms the other single-task learners to classify mode of transport. However, its superiority over RF model was not substantial. With respect to trip purpose, the multi-task classifier outperforms the single-task RF model.

One of the advantages of point-based algorithms developed in this study, such as GRU and CNN, is their autonomy from feature engineering and huge data pre-processing, which make them a more suitable tool for data analyzing tasks. This is potentially particularly important for the embedding of algorithms in mobile applications to analyze data in real-time; in this context the autonomy from data pre-processing and feature engineering would be a great advantage. Moreover, when the goal of a smartphone-based travel survey platform is to collect travel data in different cities or even in different countries, the autonomy of point-based AI algorithms from expertise knowledge make them a more flexible and transferable option. However, the trip-purpose detection algorithm in Chapter 6 still depends on some local expert knowledge, for example the distance around the destination within which the land-use and social network data are considered in the inference procedure. Developing a method to automatically find the appropriate distance within which land-use data should be considered in the modeling procedure could help a trip-purpose detection algorithm to act more independently and could result in higher prediction performance.

While the CNN or RNN algorithms may attract more attention these days, the Random Forest algorithm was also capable of predicting mode of transport or transit itinerary to a fairly high level of

accuracy, although it requires more feature engineering compared with GRU and CNN models and is more dependent on the expertise knowledge to pick appropriate features. However, the simplicity of the Random Forest model makes it a convenient and easy to understand model for everyone. Particularly, when the goal is explaining AI algorithms to people in applied, professional environments, its simplicity may be advantageous.

Such trade-offs among different models prevent the recommendation of one algorithm as the best approach. However, as shown in different studies across different fields of science, multi-task learning approach developed in this thesis deliver better performance over single-task purpose classifier. Moreover, the ensemble methods developed in this thesis shows superiority over single learners.

7.2 Current Limitations

While this thesis contributed in several directions, as explained in the previous section, there are some limitations related to it. First, although the goal of the study was to show the potential of smartphone-based travel surveys to partially or completely replace the traditional HTS, the study lacks cost-benefit analysis between smartphone travel surveys and HTS methods. While smartphone technological capabilities can contribute greatly to travel surveying by easing the burden on travelers or bringing down the total implementation cost of travel surveys, a full economic analysis can distinguish between all the advantages and disadvantages of public and private sectors involved in smartphone travel surveys. Moreover, while some studies have evaluated the total surveying costs of traditional HTS and smartphone surveys, the effort for post-processing trajectory data and developing AI algorithms should be included in cost evaluation methods to compare HTS and smartphone travel surveys accurately.

One should also consider that when applying AI algorithms on smartphone-based travel data, the quality of collected data and ground truthing methods play an important role in the performance of the learning algorithms, sometimes maybe more than the AI algorithms themselves. Hence, collecting high quality data, such as gathering high-frequency GPS data using more reliable ground truthing techniques (such as recall surveys in which travelers can modify and validate their trips

periodically with the help of a surveyor), can provide AI algorithms with richer information and affect their performance much more than the effect of methodological approach. The effect of ground truthing methods on the performance of the AI algorithms has been reported in some studies[111]. However, when considering that ground truthing method can play an important role on the quality of the labeled data, the trade-offs between the cost of ground truthing techniques, such as recall or follow-up surveys, and the performance of AI algorithms is not studied well. Such considerations will surely be important as large-scale smartphone-based surveying becomes more commonplace and requires further research to reveal which type of ground truthing methods and data quality trade-offs can and should be made.

Regarding mode detection, more work on semi-supervised methods is required to enhance their performance and take advantage of the large quantities of unlabeled data that are collected from many different sources. Other generative approaches, such as Variational Autoencoder and Decoders (VAE) may deliver better performance over GANs. Moreover, the conditional GAN method [101], which can produce samples conditioned on the class labels, may provide discriminators with more accurate samples and enhance the classification performance.

With respect to activity detection, we applied a uniform 250 meter distance around all the destinations to consider land-use and social network data. However, such distance may not be appropriate for all destinations, as in some populated urban areas there are lots of land-use data within the vicinity of a destination and in some uncrowded or rural neighborhoods there might be very few data points. Therefore, a method to find the appropriate distance around each destination could make purpose detection algorithms more flexible and independent from expertise knowledge.

The transit itinerary algorithm developed in this thesis does not include point-based features in predicting transit routes. The inclusion of such attributes may enhance the performance of transit itinerary learners, especially the closeness of GPS points to different transit routes, which may reveal more information about the chosen transit route by travelers.

Although this study has included several complementary data sources, such as land-use, GTFS, and Foursquare data, there remains a great deal of information about urban infrastructure and form that have not been included in the processing. We only considered land-use and social media data in the vicinity of destinations, while a huge amount of data that form the whole urban environment have

not been taken into account. In the next section, we suggest some directions and methodologies researchers can focus on to deal with such weakness.

7.3 Future work

The efficiency of smartphone-based travel surveys, in terms of operational costs and human resource requirements, provides motivation for further development of inference algorithms and designing higher-quality apps capable of collecting more detailed and precise GPS trajectories. Many different adaptations and experiments have been left for future work.

First, all the models developed in this thesis lack holistic information about urban infrastructure, communities, and social contexts[1] in their prediction procedure. Such holistic contextual information enables AI algorithms to recognize rich information for different neighborhoods and regions across a city. Such holistic contextual information is comparable to the knowledge of someone very familiar with the city. Usually, such an individual, when they look at a GPS trajectory over the map of a city (e.g. with Google Maps) can predict the mode of transport or even trip purpose, as his knowledge about a city's infrastructure and urban form (such as the location of metro lines and stations, bus stops, bridges, highways and freeways, popular buildings, business areas, parks, event venues, etc.) can help him to infer the mode and purpose. We can enhance the neural networks developed in Chapter 6, by providing them with such contextual knowledge. For this aim, we need a system to encode the whole contextual information of a city and feed them as input to the Neural Network model [177]. One can benefit from the geospatial indexing system developed by Google, known as S2Geometry [178, 179], or Uber, known as H3 [180]. Both systems index the earth surface to a fine grid, in Google S2 system, or hexagonal grid, in the Uber H3 system. After indexing the whole city into a grid system, we can encode detailed contextual information into the grid system and feed them as input into an embedding algorithm(s). Such contextual information could potentially play the role of large corpus of text in word embedding models, such as Word2vec[181]. The word embedding models are trained on large corpuses of text, such as Wikipedia or dictionary texts, and generate a vector space with several hundred dimensions. Such pre-trained models are

[1] Here, by context we mean "the interrelated condition in which something exists in space and time. It encompasses everything about the people, place, and circumstances of a spatial unit, be it a neighborhood, city or region." [176]

then used in many Natural Language Processing (NLP) models. It would be possible to follow the same approach to reconstruct the structural, social and community context of a city and use it in our training algorithms.

Second, the trip information identified in this thesis are among the most essential required in many transportation demand modeling applications. However, there are other attributes that could and will need to be inferred from smartphone data, especially those used in activity-based transportation demand analysis, such as tours and sub-tours or "with-whom" an individual has traveled. For example, inferring the tours from trajectory data can be accomplished by using sequential modeling approaches, such as RNN. Other trip information, such as "with-whom" an individual has completed a trip, requires further travel surveying efforts involving all household members and all the users who share a ride to participate in the travel survey.

Third, to make a strong case for fully replacing traditional household travel surveys with smartphone-based ones, one should compare them with respect to their final outputs, i.e. origin-destination(OD) matrices. It is important to evaluate the degree to which these emerging methods can compare with the traditional HTS methods on these dimensions.

Fourth, we developed all the models in this thesis based on data that had already been collected. However, real-time mode or purpose detection is an approach worth exploring, especially if the researcher aims to enhance the quality of trip validations by recommending the inferred mode of transport or trip purpose in real-time to the respondents.

Fifth, more detailed data cleaning procedures using unsupervised approaches to clustering GPS points and removing outliers and noisy data points may enhance the performance of mode detection models.

Sixth, while the performance of learning algorithms can be enhanced by predicting "target tasks", i.e. the tasks that are important for the researcher, in a multi-task framework, some studies [175] have suggested that machine learning algorithms can learn from "auxiliary tasks" too. Auxiliary tasks are labels for data not directly important for the researcher, but that have the potential to help algorithms learn target tasks better. In the context of travel data, auxiliary tasks can be any other information we asked from respondents. For example, "occupation" can play the role of an auxiliary task, instead of a feature. The assumption behind such approaches is that: the relationship between

auxiliary tasks while being trained within task-specific layers and task-specific loss functions can be better understood [175], and can help to infer target tasks more efficiently. Moreover, auxiliary tasks can be trained simultaneously inside a unique framework with target tasks and their output fed as input to the target task layers. Some studies [175] have also proposed algorithms to combine and transfer the auxiliary tasks and feed them into target tasks network, also showing a dimension for future improvement and work.

Overall, while more effort is required in the future to improve the performance of inference algorithms to detect trip information from trajectory data, the results of this research demonstrate that AI-backed smartphone-based travel survey platforms, such as Itinerum, have the potential to play an important role in future data collection for transportation demand studies. Indeed, with the rapid changes in transportation technologies, gathering mobility data is more essential than ever and smartphone-based travel surveys have the capability to respond to this need.

Besides, the future technological advances in smartphones, such as technologies deliver longer battery life or more power-efficient sensors, or more advanced wireless technologies, such as 5G that offers the possibility to enhance the positioning accuracy [182], can also benefit to the field and give more superiority to smartphone travel surveys over traditional HTS methods. Moreover, with the emergence of autonomous vehicles in the future, researchers will have access to a rich source of image and video data, recorded daily by autonomous vehicles, which can be utilized as a valuable data source in inferring trip information from trajectory data. However, how much and when such technological advances, in smartphone travel surveys and other fields of science, can lead to entirely replacing traditional HTS methods requires more research and experiments on large-scale datasets across different cities.

Bibliography

[1] Machine learning glossary — google developers, . URL https://developers.google.com/machine-learning/glossary. Accessed: 2020-04-20.

[2] J. Ortuzar and L. Willumsen. *Modelling Transport*. John Wiley and Sons, Chichester, fourth edition, 2011.

[3] Li Shen and Peter R Stopher. Review of GPS travel survey and GPS data-processing methods. *Transport Reviews*, 34(3):316–334, 2014.

[4] Jun Wu, Chengsheng Jiang, Douglas Houston, Dean Baker, and Ralph Delfino. Automated time activity classification based on global positioning system GPS tracking data. *Environmental Health*, 10(1):101, 2011.

[5] Antonin Danalet and Nicole A Mathys. The potential of smartphone data for national travel surveys. In *17th Swiss transport research conference, Monte Verità/Ascona*, pages 17–19, 2017.

[6] Jean Louise Wolf. *Using GPS data loggers to replace travel diaries in the collection of travel data*. PhD thesis, School of Civil and Environmental Engineering, Georgia Institute of Technology, 2000.

[7] Francisco Pereira, Carlos Carrion, Fang Zhao, Caitlin D Cottrill, Chris Zegras, and Moshe Ben-Akiva. The future mobility survey: Overview and preliminary evaluation. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 9, pages 1–9, 2013.

[8] Zachary Patterson, Kyle Fitzsimmons, Stewart Jackson, and Takeshi Mukai. Itinerum: The open smartphone travel survey platform. *SoftwareX*, 10:100230, 2019.

[9] Leah Flake, Michelle Lee, Kevin Hathaway, and Elizabeth Greene. Use of smartphone panels for viable and cost-effective GPS data collection for small and medium planning agencies. *Transportation Research Record*, 2643(1):160–165, 2017.

[10] Fang Zhao, Francisco Câmara Pereira, Rudi Ball, Youngsung Kim, Yafei Han, Christopher Zegras, and Moshe Ben-Akiva. Exploratory analysis of a smartphone-based travel survey in Singapore. *Transportation Research Record*, 2494(1):45–56, 2015.

[11] David T Hartgen and Elizabeth San Jose. Costs and trip rates of recent household travel surveys. *Hartgen Group, Charlotte, NC, USA*, 2009.

[12] Cambridge Systematics. *Travel Demand Forecasting: Parameters and Techniques*, volume 716. Transportation Research Board, 2012.

[13] Lara Montini, Sebastian Prost, Johann Schrammel, Nadine Rieser-Schüssler, and Kay W Axhausen. Comparison of travel diaries generated from smartphone data and dedicated GPS devices. *Transportation Research Procedia*, 11:227–241, 2015.

[14] Na Ta, Mei-Po Kwan, Yanwei Chai, and Zhilin Liu. Gendered space-time constraints, activity participation and household structure: A case study using a GPS-based activity survey in suburban beijing, china. *Tijdschrift voor economische en sociale geografie*, 107(5):505–521, 2016.

[15] Zhenzhen Wang, Sylvia Y He, and Yee Leung. Applying mobile phone data to travel behaviour research: A literature review. *Travel Behaviour and Society*, 11:141–155, 2018.

[16] Jerald Jariyasunant, Raja Sengupta, and Joan Walker. Overcoming battery life problems of smartphones when creating automated travel diaries. Technical Report No. UCTC-FR-2014-05, 2014.

[17] Gregory Bucci, Tom Morton, et al. Cell phone data and travel behavior research: symposium summary report. Technical report, United States. Federal Highway Administration, 2014.

[18] Zachary Patterson and Kyle Fitzsimmon. MTL Trajet. Technical Report 2017-2, Concordia University, TRIP Lab, Montreal, Canada, July 2017.

[19] Peter R Stopher, Li Shen, Wen Liu, and Asif Ahmed. The challenge of obtaining ground truth for GPS processing. *Transportation Research Procedia*, 11:206–217, 2015.

[20] Scott Krig. Ground truth data, content, metrics, and analysis. In *Computer Vision Metrics*, pages 247–271. Springer, 2016.

[21] Jean Wolf, Marcelo Oliveira, and Miriam Thompson. Impact of underreporting on mileage and travel time estimates: Results from global positioning system-enhanced household travel survey. *Transportation Research Record: Journal of the Transportation Research Board*, (1854):189–198, 2003.

[22] D Pearson. A comparison of trip determination methods in GPS-enhanced household travel surveys. In *84th annual meeting of the Transportation Research Board, Washington, DC*, 2004.

[23] John L Bowman, Mark Bradley, Joe Castiglione, and Supin L Yoder. Making advanced travel forecasting models affordable through model transferability. In *the 93rd Annual Meeting of Transportation Research Board, Washington, DC*, 2014.

[24] John L Bowman and Moshe E Ben-Akiva. Activity-based disaggregate travel demand model system with activity schedules. *Transportation Research Part A: Policy and Practice*, 35(1): 1–28, 2001.

[25] Patricia L Mokhtarian and Ilan Salomon. How derived is the demand for travel? some conceptual and measurement considerations. *Transportation Research Part A*, 35(695):719, 2001.

[26] Michael Patriksson. *The traffic assignment problem: models and methods*. Courier Dover Publications, 2015.

[27] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching

for low-sampling-rate GPS trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361. ACM, 2009.

[28] Ehsan Mazloumi, Mahmoud Mesbah, Avi Ceder, Sara Moridpour, and Graham Currie. Efficient transit schedule design of timing points: a comparison of ant colony and genetic algorithms. *Transportation Research Part B: Methodological*, 46(1):217–234, 2012.

[29] Jerry CN Ng and Paul M Sarjeant. Use of direct data entry for travel surveys. *Transportation Research Record*, (1412), 1993.

[30] K Habib, Joffre Swait, and Sarah Salem. Investigating structural changes in commuting mode choice preferences with repeated cross-sectional travel survey data: the contexts of Greater Toronto and Hamilton (GTHA) area. In *13th International Conference on Travel Behaviour Research, Toronto*, volume 1520, 2012.

[31] Ying Long and Jean-Claude Thill. Combining smart card data and household travel survey to analyze jobs–housing relationships in beijing. *Computers, Environment and Urban Systems*, 53:19–35, 2015.

[32] Selmer Bringsjord and Naveen Sundar Govindarajulu. Artificial intelligence. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2020 edition, 2020.

[33] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press, Cambridge, 2016.

[34] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from AlexNet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[35] Rina Dechter. *Learning while searching in constraint-satisfaction problems*. University of California, Computer Science Department, Cognitive Systems., 1986.

[36] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[37] Mohsen Rezaie, Zachary Patterson, Jia Yuan Yu, and Ali Yazdizadeh. Semi-supervised travel mode detection from smartphone data. In *2017 International Smart Cities Conference (ISC2)*, pages 1–8. IEEE, 2017.

[38] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2013.

[39] Sina Dabiri and Kevin Heaslip. Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transportation Research Part C: Emerging Technologies*, 86: 360–371, 2018.

[40] Zhibin Xiao, Yang Wang, Kun Fu, and Fan Wu. Identifying different transportation modes from trajectory data using tree-based ensemble classifiers. *ISPRS International Journal of Geo-Information*, 6(2):57, 2017.

[41] Mogeng Yin, Madeleine Sheehan, Sidney Feygin, Jean-François Paiement, and Alexei Pozdnoukhov. A generative model of urban activities from cellular data. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1682–1696, 2017.

[42] Leon Stenneth, Ouri Wolfson, Philip S Yu, and Bo Xu. Transportation mode detection using mobile phones and GIS information. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 54–63. ACM, 2011.

[43] Tao Feng and Harry JP Timmermans. Transportation mode recognition using GPS and accelerometer data. *Transportation Research Part C: Emerging Technologies*, 37:118–130, 2013.

[44] Hamid Reza Eftekhari and Mehdi Ghatee. An inference engine for smartphones to preprocess data and detect stationary and transportation modes. *Transportation Research Part C: Emerging Technologies*, 69:313–327, 2016.

[45] Arash Jahangiri and Hesham Rakha. Developing a support vector machine (SVM) classifier for transportation mode identification by using mobile phone sensor data. In *93rd Annual Meeting of Transportation Research Board*, number 14-1442, 2014.

[46] Peter R Stopher. The travel survey toolkit: where to from here? In *Transport survey methods: Keeping up with a changing world*, pages 15–46. Emerald Group Publishing Limited, 2009.

[47] Wendy Bohte and Kees Maat. Deriving and validating trip purposes and travel modes for multi-day GPS-based travel surveys: A large-scale application in the Netherlands. *Transportation Research Part C: Emerging Technologies*, 17(3):285–297, 2009.

[48] Filip Biljecki, Hugo Ledoux, and Peter van Oosterom. Transportation mode-based segmentation and classification of movement trajectories. *International Journal of Geographical Information Science*, 27(2):385–407, 2013.

[49] Nadine Schuessler and Kay Axhausen. Processing raw data from global positioning systems without additional information. *Transportation Research Record: Journal of the Transportation Research Board*, (2105):28–36, 2009.

[50] Ron Dalumpines and Darren M Scott. Making mode detection transferable: extracting activity and travel episodes from GPS data using the multinomial logit model and python. *Transportation Planning and Technology*, 40(5):523–539, 2017.

[51] S. Dabiri, C. Lu, K. Heaslip, and C. K. Reddy. Semi-supervised deep learning approach for transportation mode identification using gps trajectory data. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):1010–1023, 2020.

[52] A Santos, N McGuckin, HY Nakamoto, D Gray, and S Liss. Summary of travel trends: 2009 national household travel survey, us department of transportation, federal highway administration. Technical report, FHWA-PL-11022, Jun, 2011.

[53] L'enquête origine-destination 2018. Technical report, l'Autorité régionale de transport métropolitain, 2019. URL https://www.artm.quebec/enqueteod/.

[54] Guangnian Xiao, Zhicai Juan, and Jingxin Gao. Travel mode detection based on neural networks and particle swarm optimization. *Information*, 6(3):522–535, 2015.

[55] Philippe Nitsche, Peter Widhalm, Simon Breuss, Norbert Brändle, and Peter Maurer. Supporting large-scale travel surveys with smartphones–a practical approach. *Transportation Research Part C: Emerging Technologies*, 43:212–221, 2014.

[56] Bao Wang, Linjie Gao, and Zhicai Juan. Travel mode detection using GPS data and socioeconomic attributes based on a random forest classifier. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1547–1558, 2018.

[57] Behrang Assemi, Hamid Safi, Mahmoud Mesbah, and Luis Ferreira. Developing and validating a statistical model for travel mode identification on smartphones. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1920–1931, 2016.

[58] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Akihisa Kawanobe. Deep feature extraction from trajectories for transportation mode estimation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 54–66. Springer, 2016.

[59] Paola A Gonzalez, Jeremy S Weinstein, Sean J Barbeau, Miguel A Labrador, Philip L Winters, Nevine L Georggi, and R Perez. Automating mode detection for travel behaviour analysis by using global positioning systems-enabled mobile phones and neural networks. *IET Intelligent Transport Systems*, 4(1):37–49, 2010.

[60] Fei Yang, Zhenxing Yao, and Peter J Jin. GPS and acceleration data in multimode trip data recognition based on wavelet transform modulus maximum algorithm. *Transportation Research Record*, 2526(1):90–98, 2015.

[61] Young-Ji Byon and Steve Liang. Real-time transportation mode detection using smartphones and artificial neural networks: Performance comparisons between smartphones and conventional global positioning system sensors. *Journal of Intelligent Transportation Systems*, 18 (3):264–272, 2014.

[62] Guangnian Xiao, Juan Zhicai, and Gao Jingxin. Inferring trip ends from GPS data based on smartphones in shanghai. In *94th Annual Meeting of Transportation Research Board*, number 15-2454, 2015.

[63] Zahra Ansari Lari and Amir Golroo. Automated transportation mode detection using smart phone applications via machine learning: Case study mega city of Tehran. In *Proceedings of 94th Annual Meeting of Transportation Research Board, Washington, DC, USA*, pages 11–15, 2015.

[64] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on GPS data. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 312–321. ACM, 2008.

[65] Lijuan Zhang, Sagi Dalyot, Daniel Eggert, and Monika Sester. Multi-stage approach to travel-mode segmentation and classification of GPS traces. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences:[Geospatial Data Infrastructure: From Data Acquisition And Updating To Smarter Services] 38-4 (2011), Nr. W25*, 38(W25):87–93, 2011.

[66] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):13, 2010.

[67] Elton F de S Soares, Kate Revoredo, Fernanda Baião, Carlos A de MS Quintella, and Carlos Alberto V Campos. A combined solution for real-time travel mode detection and trip purpose prediction. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4655–4664, 2019.

[68] Joseph Broach, Jennifer Dill, and Nathan Winslow McNeil. Travel mode imputation using GPS and accelerometer data from a multi-day travel survey. *Journal of Transport Geography*, 78:194–204, 2019.

[69] Lei Gong, Takayuki Morikawa, Toshiyuki Yamamoto, and Hitomi Sato. Deriving personal

trip data from GPS data: a literature review on the existing methodologies. *Procedia-Social and Behavioral Sciences*, 138:557–565, 2014.

[70] Lei GONG, Toshiyuki Yamamoto, and Takayuki Morikawa. Comparison of activity type identification from mobile phone GPS data using various machine learning methods. *Asian Transport Studies*, 4(1):114–128, 2016.

[71] Jean Wolf, Randall Guensler, and William Bachman. Elimination of the travel diary: Experiment to derive trip purpose from global positioning system travel data. *Transportation Research Record: Journal of the Transportation Research Board*, (1768):125–134, 2001.

[72] Huiying Zhao, Dalin Qian, Ying Lv, Bo Zhang, and Rongyu Liang. Development of a global positioning system data-based trip-purpose inference method for hazardous materials transportation management. *Journal of Intelligent Transportation Systems*, pages 1–16, 2019.

[73] Kay W Axhausen, Stefan Schönfelder, J Wolf, M Oliveira, and Ute Samaga. 80 weeks of GPS-traces: approaches to enriching the trip information. *Arbeitsberichte Verkehrs-und Raumplanung*, 178, 2003.

[74] Patrick McGowen and Michael McNally. Evaluating the potential to predict activity types from gps and GIS data. In *86th Annual Meeting of Transportation Research Board*, number 07-3199, 2007.

[75] Zhongwei Deng and Minhe Ji. Deriving rules for trip purpose identification from GPS travel survey data and land use data: A machine learning approach. In *Traffic and Transportation Studies 2010*, pages 768–777. 2010.

[76] Marcelo Oliveira, Peter Vovsha, Jean Wolf, and Michael Mitchell. Evaluation of two methods for identifying trip purpose in GPS-based household travel surveys. *Transportation Research Record: Journal of the Transportation Research Board*, (2405):33–41, 2014.

[77] Youngsung Kim, Francisco C Pereira, Fang Zhao, Ajinkya Ghorpade, P Christopher Zegras, and Moshe Ben-Akiva. Activity recognition for a smartphone and web based travel survey. *arXiv preprint arXiv:1502.03634*, 2015.

[78] Zack Zhu, Ulf Blanke, and Gerhard Tröster. Inferring travel purpose from crowd-augmented human mobility data. In *Proceedings of the First International Conference on IoT in Urban Space*, pages 44–49. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2014.

[79] Alireza Ermagun, Yingling Fan, Julian Wolfson, Gediminas Adomavicius, and Kirti Das. Real-time trip purpose prediction using online location-based search and discovery services. *Transportation Research Part C: Emerging Technologies*, 77:96–112, 2017.

[80] Zong Fang, Lv Jian-yu, Tang Jin-jun, Wang Xiao, and Gao Fei. Identifying activities and trips with GPS data. *IET Intelligent Transport Systems*, 12(8):884–890, 2018.

[81] Foursquare for developers: Venue response. https://developer.foursquare.com/docs/responses/venue, 2017. [Accessed on June 12, 2017].

[82] Seyed Amir H Zahabi, Ajang Ajzachi, and Zachary Patterson. Transit trip itinerary inference with gtfs and smartphone data. *Transportation Research Record*, 2652(1):59–69, 2017.

[83] Filipe Rodrigues, Kristian Henrickson, and Francisco C Pereira. Multi-output gaussian processes for crowdsourced traffic data imputation. *IEEE Transactions on Intelligent Transportation Systems*, 20(2):594–603, 2018.

[84] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[85] Kellie J Archer and Ryan V Kimes. Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, 52(4):2249–2260, 2008.

[86] Thomas G Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2): 139–157, 2000.

[87] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/.

[88] Leo Breiman and Adele Cutler. *Random forests-classification description*, 2007. URL https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. [Accessed on April 23, 2020].

[89] Leo Breiman. Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, 1:58, 2002.

[90] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[91] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[92] Daniel Quang and Xiaohui Xie. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, 44(11): e107, 2016.

[93] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[94] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[95] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[96] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[97] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 3:1681–1690, 2016.

[98] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[99] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[100] Jost Tobias Springenberg. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. *arXiv preprint arXiv:1511.06390*, 2015.

[101] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014.

[102] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv preprint arXiv:1701.00160*, 2016.

[103] Melvin Wong, Bilal Farooq, and Guillaume-Alexandre Bilodeau. Discriminative conditional restricted boltzmann machine for discrete choice and latent variable modelling. *Journal of choice modelling*, 29:152–168, 2018.

[104] Caitlin Cottrill, Francisco Pereira, Fang Zhao, Iněs Dias, Hock Lim, Moshe Ben-Akiva, and P Zegras. Future mobility survey: Experience in developing a smartphone-based travel survey in Singapore. *Transportation Research Record: Journal of the Transportation Research Board*, (2354):59–67, 2013.

[105] P Stopher, Y Zhang, J Zhang, and B Halling. Results of an evaluation of travelsmart in South Australia. In *Australasian Transport Research Forum (ATRF), 32nd, 2009, Auckland, New Zealand*, volume 32, 2009.

[106] Heping Zhang and Burton Singer. *Recursive partitioning and applications*. Springer Science & Business Media, 2010.

[107] Lara Montini, Nadine Rieser-Schüssler, Andreas Horni, and Kay W Axhausen. Trip purpose identification from GPS tracks. *Transportation Research Record*, 2405(1):16–23, 2014.

[108] Martin Catala et al. Expanding the google transit feed specification to support operations and planning [summary]. Technical report, Florida. Dept. of Transportation. Research Center, 2011.

[109] Zachary Patterson and Kyle Fitzsimmons. Datamobile: Smartphone travel survey experiment. *Transportation Research Record: Journal of the Transportation Research Board*, (2594):35–43, 2016.

[110] Fang Zhao, Ajinkya Ghorpade, Francisco Câmara Pereira, Christopher Zegras, and Moshe Ben-Akiva. Stop detection in smartphone-based travel surveys. *Transportation Research Procedia*, 11:218–226, 2015.

[111] Guangnian Xiao, Zhicai Juan, and Chunqin Zhang. Detecting trip purposes from smartphone-based travel surveys with artificial neural networks and particle swarm optimization. *Transportation Research Part C: Emerging Technologies*, 71:447–463, 2016.

[112] Elizabeth Greene, Leah Flake, Kevin Hathaway, and Michael Geilich. A seven-day smartphone-based GPS household travel survey in indiana. In *95th Annual Meeting of Transportation Research Board*, number 16-6274, 2016.

[113] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[114] Milad Ghasri, Taha Hossein Rashidi, and S Travis Waller. Developing a disaggregate travel demand system of models using data mining techniques. *Transportation Research Part A: Policy and Practice*, 105:138–153, 2017.

[115] Chaoran Zhou, Hongfei Jia, Zhicai Juan, Xuemei Fu, and Guangnian Xiao. A data-driven method for trip ends identification using large-scale smartphone-based GPS tracking data. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):2096–2110, 2017.

[116] Muhammad Awais Shafique and Eiji Hato. Use of acceleration data for transportation mode prediction. *Transportation*, 42(1):163–188, 2015.

[117] Muhammad Awais Shafique and Eiji Hato. Travel mode detection with varying smartphone data collection frequencies. *Sensors*, 16(5):716, 2016.

[118] Thanos Bantis and James Haworth. Who you are is how you travel: A framework for transportation mode detection using individual and environmental characteristics. *Transportation Research Part C: Emerging Technologies*, 80:286–309, 2017.

[119] Ziheng Lin, Mogeng Yin, Sidney Feygin, Madeleine Sheehan, Jean-Francois Paiement, and Alexei Pozdnoukhov. Deep generative models of urban mobility. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[120] Timothy Sohn, Alex Varshavsky, Anthony LaMarca, Mike Y Chen, Tanzeem Choudhury, Ian Smith, Sunny Consolvo, Jeffrey Hightower, William G Griswold, and Eyal De Lara. Mobility detection using everyday GSM traces. In *International Conference on Ubiquitous Computing*, pages 212–224. Springer, 2006.

[121] Linlin Wu, Biao Yang, and Peng Jing. Travel mode detection based on GPS raw data collected by smartphones: a systematic review of the existing methodologies. *Information*, 7(4):67, 2016.

[122] Zhanbo Sun and Xuegang Jeff Ban. Vehicle classification using GPS data. *Transportation Research Part C: Emerging Technologies*, 37:102–117, 2013.

[123] Hao Wang, GaoJun Liu, Jianyong Duan, and Lei Zhang. Detecting transportation modes using deep neural network. *IEICE Transactions on Information and Systems*, 100(5):1132–1135, 2017.

[124] Taha H Rashidi, Alireza Abbasi, Mojtaba Maghrebi, Samiul Hasan, and Travis S Waller. Exploring the capacity of social media data for modelling travel behaviour: Opportunities and challenges. *Transportation Research Part C: Emerging Technologies*, 75:197–211, 2017.

[125] Defu Lian and Xing Xie. Collaborative activity recognition via check-in history. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 45–48. ACM, 2011.

[126] Samiul Hasan and Satish V Ukkusuri. Urban activity pattern classification using topic models from online geo-location data. *Transportation Research Part C: Emerging Technologies*, 44: 363–381, 2014.

[127] Samiul Hasan and Satish V Ukkusuri. Location contexts of user check-ins to model urban geo life-style patterns. *PLoS One*, 10(5):e0124819, 2015.

[128] Jae Hyun Lee, Adam W Davis, Seo Youn Yoon, and Konstadinos G Goulias. Activity space estimation with longitudinal observations of social media data. *Transportation*, 43(6):955–977, 2016.

[129] Zhenhua Zhang, Qing He, and Shanjiang Zhu. Potentials of using social media to infer the longitudinal travel behavior: A sequential model-based clustering method. *Transportation Research Part C: Emerging Technologies*, 85:396–414, 2017.

[130] Yongmei Lu and Yu Liu. Pervasive location acquisition technologies: Opportunities and challenges for geospatial studies. *Computers, Environment and Urban Systems*, 36(2):105–108, 2012.

[131] OpenTripPlanner Bibliography. http://docs.opentripplanner.org/en/latest/Bibliography/, 2017. [Accessed on July 18, 2017].

[132] Ville de Montreal. MTL Trajet. https://ville.montreal.qc.ca/mtltrajet/en/, 2017. [Accessed on June 15, 2017].

[133] MAMROT. Localisation des immeubles 2011. Ministere des Affaires Municipales, des Regions et de l'Occupations du Territoire (MAMROT), 2011.

[134] Meredith Kimberly Cebelak. *Transportation planning via location-based social networking data: exploring many-to-many connections*. PhD thesis, University of Texas at Austin, August 2015.

[135] Foursquare for Developers: Venue Categories. https://developer.foursquare.com/docs/venues/categories, 2017. [Accessed on June 12, 2017].

[136] Get Elevations - MSDN. https://msdn.microsoft.com/en-us/library/jj158961.aspx, 2017. [Accessed on October 20, 2017].

[137] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL http://www.R-project.org/.

[138] S Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley*, pages 225–227, 1990.

[139] Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[140] Bradley S. Stewart and Chelsea C. White. Multiobjective a*. *J. ACM*, 38(4):775–814, October 1991. ISSN 0004-5411.

[141] J. L. P´erez De la Cruz, J. L. P´erez De la Cruz, L. Mandow, and L. Mandow. A new approach to multiobjective a* search. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, page 218–223, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[142] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[143] Charu C Aggarwal. *Data mining: the textbook*. Springer, 2015.

[144] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li. *Geolife GPS trajectory dataset - User Guide*, 2011. URL https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/.

[145] Yoshua Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

[146] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354 – 377, 2018. ISSN 0031-3203.

[147] Mostafa Elhoushi, Jacques Georgy, Aboelmagd Noureldin, and Michael J Korenberg. A survey on approaches of motion mode recognition using sensors. *IEEE Transactions on Intelligent Transportation Systems*, 18(7):1662–1686, 2017.

[148] German Castignani, Raphaël Frank, and Thomas Engel. Driver behavior profiling using smartphones. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 552–557. IEEE, 2013.

[149] Cheng Ju, Aurélien Bibaut, and Mark van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818, 2018.

[150] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[151] Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems*, pages 402–408, 2001.

[152] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360, 2015.

[153] Ali Yazdizadeh, Zachary Patterson, and Bilal Farooq. An automated approach from gps traces to complete trip information. *International Journal of Transportation Science and Technology*, 8(1):82 – 100, 2019. ISSN 2046-0430.

[154] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[155] Arash Kalatian and Bilal Farooq. Mobility mode detection using wifi signals. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–7. IEEE, 2018.

[156] Augustus Odena. Semi-supervised learning with Generative Adversarial Networks. *arXiv preprint arXiv:1606.01583*, 2016.

[157] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*, 2015.

[158] Mohamed Zaki, Tarek Sayed, and Khaled Shaaban. Use of drivers' jerk profiles in computer vision-based traffic safety evaluations. *Transportation Research Record: Journal of the Transportation Research Board*, (2434):103–112, 2014.

[159] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 437–446, 2015.

[160] David Warde-Farley and Ian Goodfellow. *11 adversarial perturbations of deep neural networks*. Perturbations, Optimization, and Statistics, MIT Press, 2016.

[161] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30 th International Conference on Machine Learning, Atlanta, Georgia, USA*, volume 30, page 3, 2013.

[162] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[163] Tatjana Chavdarova and François Fleuret. SGAN: An alternative training of generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9407–9415, 2018.

[164] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[165] Rich Caruana. *Multitask Learning*, pages 95–133. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5529-2.

[166] A. Yazdizadeh, Z. Patterson, and B. Farooq. Ensemble Convolutional Neural Networks for Mode Inference in Smartphone Travel Survey. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–8, 2019.

[167] Toan H Vu, Le Dung, and Jia-Ching Wang. Transportation mode detection on mobile devices using recurrent nets. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 392–396. ACM, 2016.

[168] Sebastian Otte, Dirk Krechel, Marcus Liwicki, and Andreas Dengel. Local feature based online mode detection with recurrent neural networks. In *2012 International Conference on Frontiers in Handwriting Recognition*, pages 533–537. IEEE, 2012.

[169] Matteo Simoncini, Leonardo Taccari, Francesco Sambo, Luca Bravi, Samuele Salti, and Alessandro Lori. Vehicle classification from low-frequency GPS data with recurrent neural networks. *Transportation Research Part C: Emerging Technologies*, 91:176–191, 2018.

[170] Hongbin Liu and Ickjai Lee. End-to-end trajectory transportation mode classification using Bi-LSTM recurrent neural network. In *12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 1–5. IEEE, 2017.

[171] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[172] Rahul Dey and Fathi M Salemt. Gate-variants of Gated Recurrent Unit (GRU) Neural Networks. In *IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[173] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of ated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[174] Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

[175] Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. *arXiv preprint arXiv:1802.09913*, 2018.

[176] Leigh Lane, Ann Hartell, Teresa Townsend, and Ann Steedly. Defining community context in transportation project planning and development process. Technical Report of NCHRP Project 25-25 Task 69, 2011.

[177] Godwin Badu-Marfo, Bilal Farooq, and Zachary Patterson. Perturbation methods for protection of sensitive location data: Smartphone travel survey case study. *Transportation Research Record*, page 0361198119855999, 2019.

[178] Varun Pandey, Andreas Kipf, Dimitri Vorona, Tobias Mühlbauer, Thomas Neumann, and Alfons Kemper. High-performance geospatial analytics in hyperspace. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2145–2148. ACM, 2016.

[179] S2 Geometry Library. https://github.com/google/s2geometry, . Accessed: 2019-12-01.

[180] Uber's Hexagonal Hierarchical Spatial Index. https://eng.uber.com/h3/. Accessed: 2019-12-01.

[181] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[182] Ying Liu, Xiufang Shi, Shibo He, and Zhiguo Shi. Prospective positioning architecture and technologies in 5G networks. *IEEE Network*, 31(6):115–121, 2017.

[183] ZF Li, L Yu, Y Gao, Y Wu, D Gong, and G Song. Extraction method of temporal and spatial characteristics of residents' trips based on cellular signaling data. *Transportation Research*, 2(1):51–57, 2016.

[184] D Mitchell. New traffic data sources–an overview. Technical report, Bureau of Infrastructure, Transport and Regional Economics, Canberra, ACT, Australia, 2014.

[185] Jie Yang and Yingying Chen. Indoor localization using improved RSS-based lateration methods. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6. IEEE, 2009.

[186] Sudarshan S Chawathe. Beacon placement for indoor localization using Bluetooth. In *11th International IEEE Conference on Intelligent Transportation Systems*, pages 980–985. IEEE, 2008.

[187] Stephen Greaves, Adrian Ellison, Richard Ellison, Dean Rance, Chris Standen, Chris Rissel, and Melanie Crane. A web-based diary and companion smartphone app for travel/activity surveys. *Transportation Research Procedia*, 11:297–310, 2015.

[188] AJ Richardson, ES Ampt, and AH Meyburg. Nonresponse issues in household travel surveys. In *Transportation Research Board Conference Proceedings*, volume 10, pages 79–114, 1996.

[189] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[190] Xiangang Li and Xihong Wu. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4520–4524. IEEE, 2015.

[191] Yoshua Bengio et al. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[192] Abdel-rahman Mohamed, George E Dahl, Geoffrey Hinton, et al. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech & Language Processing*, 20(1): 14–22, 2012.

[193] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

[194] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[195] Ian Goodfellow, Honglak Lee, Quoc V Le, Andrew Saxe, and Andrew Y Ng. Measuring invariances in deep networks. In *Advances in Neural Information Processing Systems*, pages 646–654, 2009.

[196] Jianshu Chen and Li Deng. A new method for learning deep recurrent neural networks. *arXiv preprint arXiv:1311.6091*, 2013.

[197] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2, 2012.

[198] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[199] Tapani Raiko, Harri Valpola, and Yann LeCun. Deep learning made easier by linear transformations in perceptrons. In *Artificial Intelligence and Statistics*, pages 924–932, 2012.

[200] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *31st International Conference on Machine Learning (ICML)*, 2014.

[201] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[202] Vladimír Boža, Broňa Brejová, and Tomáš Vinař. DeepNano: deep recurrent neural networks for base calling in MinION nanopore reads. *PLoS One*, 12(6):e0178751, 2017.

[203] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[204] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[205] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[206] Jordan B Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105, 1990.

[207] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136, 2011.

[208] Léon Bottou. From machine learning to machine reasoning. *Machine Learning*, 94(2):133–149, 2014.

[209] Tsungnan Lin, Bill G Horne, Peter Tino, and C Lee Giles. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

[210] Michael C Mozer. Induction of multiscale temporal structure. In *Advances in Neural Information Processing Systems*, pages 275–282, 1992.

[211] Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*, pages 493–499, 1996.

[212] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[213] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[214] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

[215] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[216] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation

with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[217] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[218] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552, 2009.

[219] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

[220] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[221] Ahmed Elsheikh, Soumaya Yacout, and Mohamed-Salah Ouali. Bidirectional handshaking lstm for remaining useful life prediction. *Neurocomputing*, 323:148 – 156, 2019. ISSN 0925-2312.

[222] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney. A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2462–2466. IEEE, 2017.

Appendix A

Mobile Technologies for Mobility Data Collection

This chapter includes more detailed information on the following issues:

- Mobile Technologies for Mobility Data Collection
- Ground Truthing Methods and Technologies

A.1 Mobile Technologies for Mobility Data Collection

Current section covers different data collection technologies in mobile phones, as shown in Figure 1.1 applicable in travel surveying.

Cellular networks locate a mobile user based on measured radio signals from a cellular tower [183]. Such locational data is gathered automatically by mobile network providers, without any burden on the users[15]. Also, there is no need for any mobile application being installed on the users' phones, as the data is gathered by phone network provider not by the mobile phone of the users. However, as the spatial information in cellular networks is estimated and gathered by cellular towers, their spatial accuracy is lower compared to other techniques such as GPS positioning. Such cellular networks can gathered locational data with accuracy varies between 100 and 10,000 meters [183].

GPS positioning is a system of embedded GPS sensors which locate the mobile phone devices by

signals transmitted by satellites. The GPS positioning can record high spatial resolution data, with few meters accuracy. However, gathering such locational data requires installing an application on users' mobile phones. It should be mentioned that recording trajectories of travelers can also be carried out via GPS devices [73]. Such studies requires travelers to bring a GPS device every where they go, which is not applicable in large-scale HTS studies and put a lot burden on the participants. However, GPS positioning system, in general, suffers from the signal lost, specially in urban areas, which reduce the accuracy of recorded data [184].

WiFi positioning technique locates the mobile phone devices using embedded wireless and WiFi access points. It can detect the position of mobile phones based on the strength of received signal, using algorithms known as RSSI (Received Signal Strength Indication) methods[185]. The spatial resolution of WiFi positioning technique is higher than cellular network, but is not always as accurate as GPS positioning. However, the spatial resolution can be enhanced to few meters with applying some positioning techniques [15]. The other disadvantage of WiFi positioning technique is that the WiFi access points are scarce in suburbs or on the highways or roads far from residential or businesses buildings, which prevent collecting data over large urban areas.

Bluetooth positioning locates the Bluetooth enabled smartphones whenever they pass through places with Bluetooth beacon [1]. The advantage of such technology is that most mobile phones are equipped with Bluetooth today. However, in many cases, the location is recorded without any timestamp and it can only cover devices within a certain distance (normally 100 meters) [15].

Regarding the motion systems, accelerometer is the most well-know sensor in today's smartphones that have been widely used in many activity recognition studies. It enables us to measure the acceleration and motion status of smartphone user. Digital compass or Magnetometer sensor detects the orientation of the mobile device relative to the magnetic earth. Gyroscope is another sensor in mobile phones that records changes in orientation or, in some devices, it can detect the changes in rotational velocity [15].

[1] A beacon is a small Bluetooth radio transmitter that constantly transmits Bluetooth signals. It can be used to broadcast advertisements and notifications to the Bluetooth enabled smartphones in their vicinity [186]

While each of the above mentioned technologies possess their own advantages, we should also consider other issues when considering them in a travel survey context. The most important considerations are the resolution and the accessibility of the data, and the effect of data collection technique on mobile phone battery usage. The cellular data gathered by network providers is obtainable through specific agreements and contracts, which potentially restricts their usage in HTS studies due to privacy issues[17]. Moreover, in many urban areas the resolution of cellular data is low. Bluetooth technology is restricted to very short distances and cannot be applied on large distance trips. WiFi positioning system also is not accurate enough, specially in areas with sparse or no WiFi coverage. However, GPS positioning can locate the mobile phone devices with enough accuracy, usually with a 5-meter accuracy[15]. Moreover, the GPS data can be easily collected by installing an application on the smartphones, which record the GPS trajectory of travelers with timestamp. Furthermore, the accuracy of data can be improved if the GPS information is recorded with high frequency, for example every 5 or 1 second. Indeed, there is a trade-off between the quality of collected data and the battery usage. Recording the GPS positioning information with high frequency or using motion sensors, such as accelerometer or gyroscope, may considerably increase the smartphones' battery usage, and may not be applicable on large-scale and long-term data collection surveys.

A.2 Ground Truthing Methods and Technologies

Ground truthing in GPS trajectory data means labeling the trajectory with valuable trip information, such as mode(s) of transport. Such labels are usually obtainable by asking respondents to validate their trip information through three techniques: in-app prompted technique [8], web-based validation technique [110, 187], and recall (sometime referred to as follow-up or surveyor-intervened) technique [10]. The in-app prompted technique is achieved via asking respondent to validate their trip information inside the mobile phone application. Some applications, such as Itinerum[8], enable researchers to prompt the users and ask them some questions upon detecting a stop. The goal of such questions is mainly gathering the ground truth data.

The recall validation technique is another method where the traveler will be contacted via phone call or mail to ask questions about their recorded trajectories. Such validation technique usually

requires human resources and increase the surveying costs.

In the web-based validation technique, the respondents are asked to log into a website and validate their trip at the end of the day. The trajectories usually are shown to the travelers on the map [187] and travelers can review and validate his trip information on the map or through a series of questions. While there are different validation methods in the literature, each of them have their own advantages and disadvantages. The in-app validation is the most efficient methods in terms of human resource requirements and implementation costs, as there is no need to hire surveyors like the follow-up technique, or design and develop a website in web-based validation technique. It also does not suffer from the forgetfulness of travelers at the end of the day, which is a well-known disadvantage of recall surveys or traditional HTS methods [188]. The recall techniques places more burden on the travelers, sometimes as much as the traditional HTS methods. However, if we do not consider the implementation and human resource costs and the amount of burden on travelers, the best validation method is the combination of in-app validation with recall or web-based techniques. For example, Xiao et al. [111], have implemented surveyor-intervened prompted recall surveys to enhance the accuracy of validations. Next section introduces the Itinerum, which is the platform used to gather travel data in current study.

Appendix B

Complementary Data

This Appendix includes more detailed information on the following issues:

- Land-use data
- Foursquare data

B.1 Montreal Land-use Data

Land-use information is one of the data sources from which the activities around a GPS coordinate can be derived. The last updated Montreal land-use data is collected by MAMROT [1] in 2011. Montreal land-use data contains land-use characteristics at the parcel level for the Montreal Metropolitan Area. There are around 970 land-use types which have been classified into 23 categories. As shown in Table B.1, a great part of land-use parcels are residential buildings. Hence, using just land-use data may cause the trip purpose detection algorithm to be prone to classification error, due to myriad residential parcels around trip destinations. Thus, we sought other location based data sources, such as Foursquare, to use them as a complement to land-use data.

[1] Ministere des Affaires Municipales, des Regions et de l'Occupations du Territoire

Table B.1: Land-use Categories and Their Frequency in Montreal Metropolitan Area

ID	Category	Frequency
1	Residential	977,302
2	Unoccupied	83,215
3	Public Ways	26,001
4	Malls and Shopping Centres	8,948
5	Agriculture-Forest	8,715
6	Business & Services	6,844
7	Industry	5,500
8	Recreational	3,992
9	General Goods Retailers	3,012
10	Governmental Buildings	2,596
11	Hotels & Accommodation	2,480
12	Health-Related Buildings	1,722
13	Transportation Infrastructures (Motor Vehicle)	1,619
14	Parking	1,523
15	Public Services	1,509
16	Educational	1,410
17	Food Retailers	749
18	Wholesale	708
19	Freight Service & Railways Infrastructures	568
20	Cloth & Furniture Retailers	360
21	Transportation Infrastructures (Aviation)	305
22	Construction Material Retailers	255
23	Transportation Infrastructures (Maritime)	134

B.2 Foursquare Data

Foursquare is an online location-based social network through which individuals can connect with the places they visit by 'check-ins' using the Foursquare app. In general, a check-in specifies that a certain user has been presented at a certain venue. The check-ins then is shared with the venue as well as the friends who installs the app [134]. In current study, for each trip destination we sent a request to Foursquare API to search all the venues in 250 meter radius around a trip destination. According to the online Foursquare API documentation [81], each request to Foursquare API returns maximum of 50 venues. For each venue, there are around 35 fields of information, among which the following have been used in current study:

- Categories, which indicates the Foursquare sub- or sub-sub-category to which the venue belongs
- Stats, which contains two useful information:

  (1) *checkinsCount* which is the total check-ins ever in a venue

  (2) *usersCount* which is the total users who have ever checked in a venue

As stated in Foursquare API Documentation for Venue Categories [135] Foursquare has categorized venues into 10 top-level categories, as shown in Table B.2.

Table B.2: Foursquare Top-level Venue Categories, their Frequency, *CheckinsCount* and *UsersCount*, in our data set (Ranked by *CheckinsCount*).

id	Category name	Frequency	checkinsCount	usersCount
1	Food	9290	85,441,649	39,023,641
2	Shop & Service	14,557	69,297,192	24,410,157
3	Professional & Other Places	13,255	68,545,764	9,713,876
4	Outdoors & Recreation	4,007	35,732,827	6,889,983
5	Travel & Transport	3,985	31,126,901	6,950,913
6	College & University	1,794	29,589,063	5,434,526
7	Nightlife Spot	2,104	16,541,388	7,489,199
8	Arts & Entertainment	2,603	16,485,299	8,555,722
9	Residence	2,568	8,511,930	826,256
10	Event	23	137,351	85,447

Also, each top-level category has 'sub-' and 'sub-sub' categories which results in total 910 categories. In current study we have aggregated all *checkinsCount* for each top-level category, resulted

in 10 *checkinsCount* attribute around each trip destination. Also, the same procedure has been done for *usersCount*. This information further will be included in the random forest model to predict the trip purpose. The values of frequency, *checkinsCount* and *usersCount* for Foursquare top-level venue categories in our data set have been demonstrated in Table B.2.

Appendix C

Explanation of Mathematical Operations and Concepts of the Developed Models

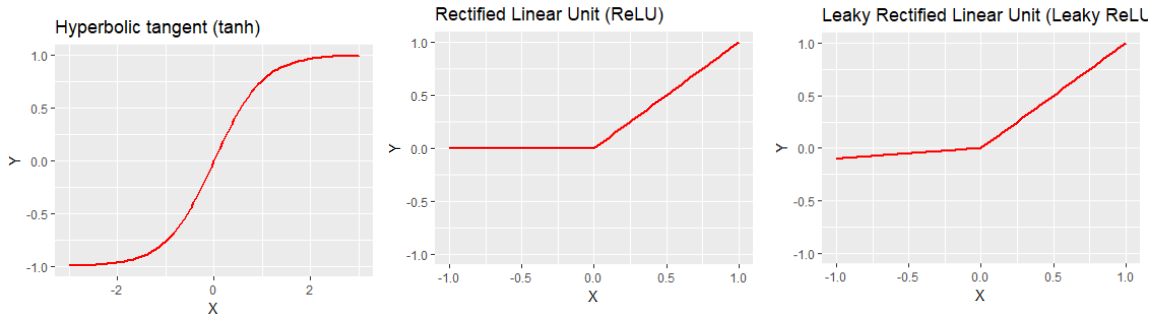
This Appendix includes more detailed information on the following modeling approaches:

- Convolutional Neural Network
- Recurrent Neural Networks
- Generative Adversarial Networks

C.1 Convolutional Neural Network

C.1.1 Non-linear Activation Functions

The second stage of a convolutional layer in a CNN model is non-linear activation. Several non-linear activation functions are used in developing neural networks, among which the Hyperbolic tangent (tanh) function, rectified linear units (ReLU), and leaky rectified linear unit (LeakyReLU) are the most well-known [33]. Figure C.1 shows the plots and equations of these functions.



(a) $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ (b) $Relu(x) = \begin{cases} 0 & for x < 0 \\ x & for x \geq 0 \end{cases}$ (c) $LRelu(x) = \begin{cases} \alpha x & for x < 0 \\ x & for x \geq 0 \end{cases}$

Figure C.1: Generally Known Non-linear Activation Functions Used in Convolutional Neural Networks

C.1.2 Pooling

Typically, after the non-linear activation function, pooling operation is applied. A pooling operation reduces the size of the output feature maps by replacing the elements of them with a summary statistic of those elements, like taking the maximum value (max pooling) [33, 90]. Pooling makes the representation of the input data approximately invariant to small translations (changes) of the input details. Invariance to translation means that if we make small changes to input data, the output of the pooling operation (the pooled output) does not change [33].

FigureC.2 shows an example of max pooling and how the invariance to local translation works. Both Figures C.2.a and C.2.b show a max pool operation with pooling window size=3 and stride=1. Figure C.2.a is a view of the middle output of a convolutional layer. The top row demonstrates the output of the non-linear activation function. The bottom row shows the outputs of the max pooling. Figure C.2.b shows a view of the same network after the values of three elements (for example the acceleration of three GPS points) have been changed. Despite the change in the three values of the top row (red circle in Figure C.2), the values of the bottom row have not changed, because the max pooling operation is sensitive only to the maximum value in a neighborhood and is approximately invariant to the changes in small values in the pooling region[33].

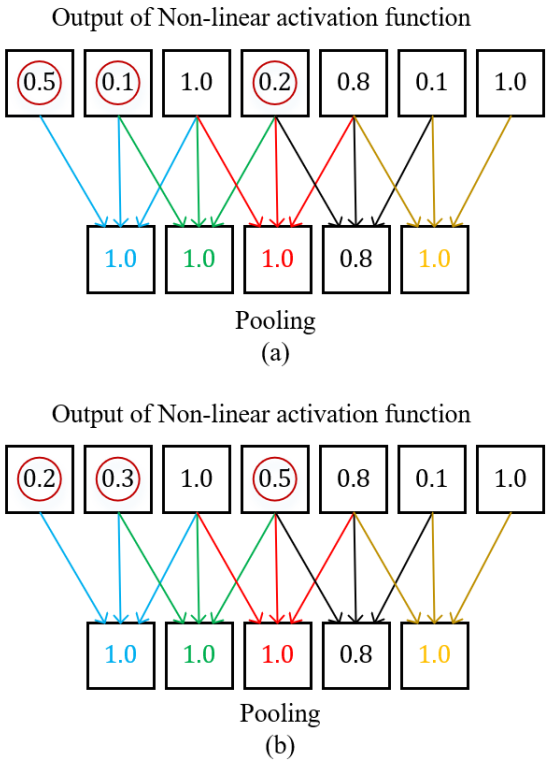


Figure C.2: Max pooling operation and how it introduces invariance (window size=1, stride=1).

C.2 Recurrent Neural Network

This section provides different architectures of Recurrent Neural Networks that have been developed across different fields of science.

C.2.1 Other Architectures of Recurrent Neural Networks

Bidirectional RNNs

The RNN models explained in the previous section, only captures the information from the past. In other words, at time step *t*, only the impact of previous sequence of inputs $(x^{(1)}, x^{(2)}, ..., x^{(t-1)}, x^{(t)})$ on $(y^{(t)}$ are captured by the recurrent neural network structure. However, there are cases where the output $(y^{(t)}$ depends on the whole sequence of inputs, i.e $(x^{(1)}, x^{(2)}, ..., x^{(T-1)}, x^{(T)})$ [33]. Bidirectional RNNs responds to that need. A Bidirectional RNN is comprised of two RNNs, one contains the sequences from the start of the sequence and moves forward through time, the other RNN, begins from the the end of the sequence and move backward through time [33].

There are three main subject areas were the bidirectional RNNs have been used:

- speech recognition
- handwriting recognition
- bioinformatics

Also, the bidirectional RNNs have been applied to other fields of study, and we have tried to cover them as much as possible.

In speech recognition, the correct interpretation of a phoneme at step t may depend not only on the previous phonemes but also on the next few phonemes. Schuster et al. [189] developed the first Bidirectional RNNs that processes data in both directions through two separate hidden layers. The two hidden layers are fed forward into the same output layer.

In handwriting recognition, the similar dependencies between nearby words in a text require a different network structure to capture the future and the past sequence of inputs at time step t. Bidirectional RNNs responds to that need in handwriting recognition.

Deep Recurrent Neural Networks

Deep and hierarchical networks can result in better prediction performance than a shallow one [190, 191]. Moreover, other researchers observed that the depth of a neural network is more important than layer size [192, 193, 194]. Experiments proved the need for enough depth to provide the required mapping in neural networks [33, 94].

RNNs consists of three sets of parameters, and their corresponding transformations, to carry out all the computations [33]:

- parameters for input-to-hidden function
- parameters for hidden-to-hidden function
- parameters for hidden-to-output function

There are lots of ways to make a recurrent neural network deep. We can add an intermediate, nonlinear hidden layer to any of the above transitions, i.e. hidden-to-hidden, input-to-hidden and hidden-to-output [193, 194]. For example, we can add intermediate nonlinear layers between two

consecutive hidden states $h^{(t-1)} \rightarrow h^{(t)}$, or between the input $x^{(t)}$ to hidden state $h^{(t)}$. In general making an RNN deep can be achieved through four ways [194]:

(1) deep input-to-hidden function

(2) deep hidden-to-output function

(3) deep hidden-to-hidden transition

(4) stack of hidden states

We briefly explain these methods.

A. Deep input-to-hidden function

Making the input-to-hidden transition deep, we produce a higher-level representation of the original input, which leads to a better job at disentangling the underlying factors of variation [194, 195]. Such higher-level representations hypothetically facilitate the learning of the temporal structure between successive time steps, as the relation between such abstract features can be revealed more easily [194]. This idea has been demonstrated by the work of Mikolov et al. [181]. Their study showed that word embedding from neural network language processing models is related to their temporal neighbors by simple algebraic relationships. Chen and Dang [196] have developed a deep recurrent neural network for speech recognition by making the input-to-hidden function deep. They used a fully connected deep neural network to compute the input sequence to the RNN. The deep neural network plays the role of a feature extractor that receives its input from raw data sequences. While their deep RNN was among the first to replace input with the extracted feature, their deep input-to-hidden function was not jointly trained with other sets of parameters of the RNN.

B. Deep hidden-to-output function

Adding more layers to hidden-to-output transition can help RNN to better predict the output [33]. We can add feedforward layers as intermediate layers between the hidden state and the output. Changing the hidden to output layers is another method to makes the RNNs deep. For example, Boulanger-Lewandowski et al. [197] developed an RNN model to discover temporal dependencies in high-dimensional sequences, such as short-term spectra in audio music. They replace the output

layer of an RNN model with a Restricted Boltzmann Machines (RBMs) to predict the conditional distribution of the next time step given previous time steps in symbolic sequences of polyphonic music [194, 197].

C. Deep hidden-to-hidden transition

In a conventional RNN the transition between the consecutive hidden states is an affine transformation followed by an element-wise nonlinearity, which is indeed a shallow operation. In some cases, such shallow operation does not allow the hidden state of an RNN to rapidly adapt quickly changing modes of the input, while still preserving a useful summary of the past [194]. Pascanue et al. [194] has argued that the procedure of producing a new hidden state, from the previous hidden state and the new input, should be highly nonlinear. Hornik et al. [198] proposed using a multi-layer perceptron(MLP) with one or more hidden layers, to model the highly nonlinearity transition. Later, Bengio et al. [171] argued that with a deep transition more gradient descent steps are required to propagate back in time. Such change may cause the training of an RNN to be more difficult with a deep transition. To address such problem, Raiko et al.[199] suggested adding shortcut connections in deep transitions. The added shortcut (skip) connections skip the intermediate layers of MLP in hidden-to-hidden transitions.

The hidden-to-hidden transition can also be replaced by other types of neural networks. For example, Pinheiro and Collobert [200] deployed convolutional neural network as the transition between consecutive hidden states. Their recurrent convolutional neural network (RCNN) demonstrated faster scene parsing and achieved the cutting-edge result in Stanford Background and SIFT Flow datasets.

Furthermore, the above three methods can be combined to make an RNN deep. Pascanu et al. [194] proposed having a separate MLP for each of the above mentioned functions, i.e. hidden-to-hidden transition, input-to-hidden function and hidden-to-output function.

D. Stack of hidden states

We can extend an RNN deeper by decomposing the hidden recurrent state of it into hierarchical recurrent hidden layers [33]. The resulted RNN referred to as stacked RNN [194], have multiple

recurrent layers that are stacked on top of each other. Each of the stacked recurrent layers operates a different timescale.

Having multiple hierarchical hidden layers in an RNN showed significant benefit in the study done by Graves et al. [201]. Their proposed stacked RNN model consists of three hidden layers, each feeding up to the layer above. Also, they used skip connections between inputs and all hidden layers, as well as skip connections from all hidden layers to the outputs. The stacked RNN was used for handwriting synthesis, which showed the ability to produce favorably realistic cursive handwriting in a vast variety of styles.

Stacked RNN is recently used to develop a DNA base caller. Boza et al. [202] used an RNN with three hidden state layers to translate the sequence of nanopores into the DNA sequence. By passing single-stranded DNA fragments through nanopores, changes in the electric current are measured. The value of electric current is highly dependent on the context of several DNA bases moving through the pore during the measurement. Depend on these changes, the sequence of electric current measurements is split into events. Afterward, the sequence of events is fed into a base caller (i.e. the stacked RNN), which translates the sequence of the events into a DNA sequence.

Encoder-Decoder Architectures

An RNN can map an input to an output in different ways:

- mapping a sequence to a fixed-size vector
- mapping a fixed-size vector to a sequence
- mapping a sequence to a sequence of the same length
- mapping a sequence to a sequence of different length

The first three ways have been explained in the previous section. Mapping a sequence to a sequence of different lengths occurs in many studies, e.g. speech recognition, machine translation, and chat-bots [33]. The first RNN model that mapped a sequence to another sequence with different length was proposed by [203] and [204] for language translation. The proposed architecture was called encoder-decoder or sequence-to-sequence architecture.

Encoder-decoder RNNs consist of two RNN models: an encoder RNN is fed by the input sequence $X = (x^{(1)}, ..., x^{(n_x)})$ and generates a representation of input context, which is a function of its final hidden state in a form of a fixed-size vector. A decoder RNN is fed by this fixed-size vector and generates the output sequence $Y = (y^{(1)}, ..., y^{(n_y)})$. It should be mentioned in this architecture n_x and n_y can be different values. By jointly training the two RNNs we maximize the log probability of *Y* given *X*, i.e. $max(logP(y^{(1)}, ..., y^{(n_y)}|x^{(1)}, ..., x^{(n_x)}))$ [33]. We can see that in this architecture there is no constraint of $n_x = n_y$, hence the encoder and decoder can have different size of hidden layers.

The only limitation of this architecture is the fixed-size of encoder vector that makes it too small to properly represent a long sequence in some cases. Bahdanau et al. [205] proposed an architecture that its encoder generates a variable-sized vector, instead of a fixed-sized one. They also introduced an attention mechanism for processing large sequences. Their attention-based architecture consists of three components:

- *reading process* reads raw data (such as word sequences) and maps them to distributed representations. In this stage, one feature vector is associated with each position in the sequence.
- *a memory* that contains the list of featured vectors obtained from the output of the *reading process*.
- *exploiting process* that exploits the memory content and puts attention on one element of memory at each time step. This component generated the output of the decoder.

Recursive Neural Networks

Recursive neural networks are a generalization of RNNs, which are structured as a tree, rather than the sequence-like (chronological) structure of RNNs. Recursive structures are usually found in different modalities. For instance, syntactic rules of a language are commonly considered as recursive. The noun phrases that contain relative clauses, that themselves consist of noun phrases, e.g. *the house which has nice balcony*. Pollack [206] first introduced recursive neural networks to natural language processing. They presented a connectionist model to develop compact distributed representations of variable-sized recursive data structures, such as trees.

Tree structures also exist in scene images that discover the proximity and part-of relationship. For example, a bus is often found on top of street regions in an image. Also, a region in an image that contains a bus can be recursively broke down into smaller bus regions characterizing some parts like windows, tires, lights, etc. A window can also be part of a church or a house, or a tire can be found in other contexts such as a car or bike region, too. Socher et al. [207] developed recursive neural networks to predict the recursive structure in multiple modalities. Their model was able to successfully merge image segments or natural language words by learning the semantic c transformations of their original features.

One challenge of developing recursive neural networks is choosing the appropriate tree structure [33]. Many studies have suggested a structure that is not dependant on the data, as a balanced binary tree [33]. Other studies suggest using external methods to choose the tree structure. For example, in natural language sentence processing, a natural language parser provides the parse tree of the sentence, which can be used as the tree structure for the recursive network. The other approach is to let the learner itself determine the appropriate tree structure for a given input data [208].

Inclusion of multiple time scales in recurrent neural networks

Recurrent neural networks can benefit from learning long-term dependencies. While a conventional RNN contains single-step connections, it is possible to create connections between units with longer delays [209]. However, gradients propagated over many time-steps (stage) tend to vanish or explode [209]. Moreover, in many cases, the weights associated with long-term interactions are exponentially smaller compared to short-term ones, due to the multiplication of many Jacobians [33]. There are several strategies in the literature to solve such problems, such as adding skip connections through time, leaky units and removal of connections. We briefly explain these ideas.

Skip connections across time were the first approach to add long-term dependencies to recurrent neural networks. They allow the RNN model to operates over multiple time scales. In this case, some parts of the model capture the small details over fine-grained time scales, and the other parts handle the information from remote past over coarse time scales. Adding skip connections, also known as direct connections, from variables in the far past to the variables in the present is the most common way to obtain coarse time scales [209]. While in a conventional RNN the connections

starts from a unit at time *t* to a unit at time *t+1*, it is possible to create connections between units with longer delays [209]. Recurrent connections with longer delays mitigate the gradient vanishing problem, however, gradients may still explode exponentially, as the model contains both single step and delayed connections [209].

The most well-known strategy to include long-term dependencies in RNNs is the leaky unit. Leaky units were first proposed by Mozer [210] and El Hihi and Bengio [211]. Leaky units integrate different time scales through *linear self-connections* with weights near one. To explain the *linear self-connections* let first explain the concept of moving average. A running average is a term in statistics that refers to an average that continuously changes as more data points are gathered. To calculate the running average, $\mu^{(t)}$, of a value $x^{(t)}$, we can apply $\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1-\alpha)x^{(t)}$. The α in this formula is an instance of a *linear self-connections* from time step *t-1* to *t* [33]. To force a linear self-connections to remember information from far past, the value of α should set near one. When α is near zero, the information from the previous time step will quickly be discarded. The hidden units with such linear self-connections are referred to as leaky units.

The main difference between skip connections and linear self-connections is in the way they convey information from past to present. While skip connections always transfer information from *d* time steps earlier (where *d* is an integer number), the linear self-connections allow smoothly transferring information from past by adjusting α value (where α is a real number). It should be mentioned that the time constants, i.e. α values, in leaky units can be set to fix value or free parameters that their value should be learned [194, 210].

Gated networks

Gated networks are considered as the most effective sequence modeling technique in practical applications [33]. Both leaky units and gated networks can accumulate information over a long duration and forget it when it is useful. However, the forgetting process, i.e. clearing the old information, in gated networks is done automatically, instead of manually deciding when to forget the disadvantageous information.

The idea behind the gated recurrent neural networks is to create paths through time, that their derivatives neither vanish nor explode [33]. Gated RNNs accomplish this by considering connection

weights that may change at each time step. The most well-known gated recurrent neural network are long short-term memory. We explain the LSTM in the following.

Hochreiter and Schmidhuber [212] firstly introduced the long short-term memory unit. Afterward, there have been several minor modifications on the LSTM unit, however, the idea of introducing internal recurrence (a self-loop), that provide long-term paths through which gradient can flow, is the main contribution of original LSTM model. While the weights on the self-loops were fixed in the original LSTM, later, Gers et al. [213] proposed to make the weight on the self-loop conditioned on the context.

The main difference between LSTM recurrent networks and conventional RNNs is that the former contains internal recurrence in addition to the outer recurrence of conventional RNNs. While each cell of an LSTM recurrent network has the same inputs and outputs of a conventional RNN, the internal recurrence controls the flow of information through additional weights and a system of gated hidden units [33]. The internal recurrence contains a linear self-loop, similar to the leaky units explained above. While the time constant in leaky units, i.e. α, is determined manually, the weights of linear self-loop of an internal recurrence are under control of a forget gate unit, which is defined for each time step t and each LSTM cell i. The forget unit is indeed a sigmoid function that maps the associated weights to a value between 0 and 1. The gated units also control the time scale of integration dynamically. Even, an LSTM with fixed parameters can change the time scale of integration based on the input sequence. These characteristics allow LSTM to be fed with different size of sequences.

LSTM have been applied in various field of study, such as image captioning [214, 215, 216], speech recognition[193], machine translation[217, 218], and parsing[219]. Deeper architecture of LSTM also have been deployed for speech recognition.

Deep LSTM

As mentioned previously deep neural networks can result in better prediction performance than a shallow one [190, 191]. Li and Wu [190] implemented a stacked LSTM architecture for a large vocabulary conversational telephone speech recognition task. They also analyzed the developed models based on three main functions used in RNN architecture, i.e. input-to-hidden function,

hidden-to-hidden transition, and hidden-to-output function.

Bidirectional LSTM

Graves and Schimhuber [220] combined Bidirectional RNNs with LSTM for framewise phone classification. Their model mapped a sequence of speech frames to a corresponding sequence of phoneme labels. The developed Bidirectional RNN can access long-range context in both input directions and outperformed both the unidirectional RNN and Long Short Term Memory (LSTM). They concluded where context is vitally important in speech processing tasks the bidirectional LSTM is a befitting architecture.

Recently, Bidirectional LSTM has been used to predict the remaining useful life of a physical system[1] [221]. In such problems, the output is not a sequence and only the value of "remaining useful life" should be predicted. The authors have proposed an architecture that processes the observed sequence in both directions sequentially, instead of processing both forward and backward sequence simultaneously. Their proposed architecture first processes the sequence in the forward direction. Afterward, it initializes processing the backward direction by using the LSTM final states. The architecture produces two different yet linked mappings of the observation sequences to the desired output value.

Deep bidirectional LSTM

Graves et al. [193] have stacked multiple forward and backward LSTM hidden layers on top of each other, where the output sequence of one layer plays the role of the input sequence for the next layer. Their deep bidirectional LSTM architecture was applied to speech recognition, and they found that it provided a noticeable improvement over single-layer LSTM.

Training deep bidirectional LSTMs is expensive in terms of time and effort to tune. Zayer et al. [222] have carried out a comprehensive study of different aspects of training LSTM and deep bidirectional LSTMs for speech recognition tasks, such as network topology, sequence chunking and batch sizes, optimization methods and regularization. They trained a series of deep bidirectional

[1] "Remaining useful life (RUL) prediction is the attempt to predict the remaining period of normal operation, at a certain level of performance, for a physical system" [221]

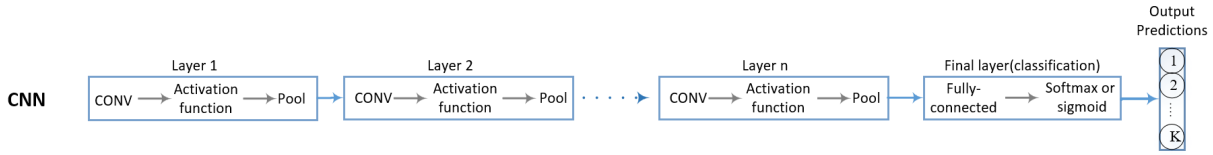
LSTM models with up to 10 layers for speech recognition. They also tuned the other aforementioned hyperparameters and noticed that the more improvement of other hyperparameters, the deeper the optimal network becomes. They also compared the Bidirectional vs. Unidirectional architectures of deep LSTM models and found that bidirectional networks outperform better than unidirectional ones. As mentioned above, such observation has been confirmed by other researchers [189, 220], too.

C.3 Generative Adversarial Networks

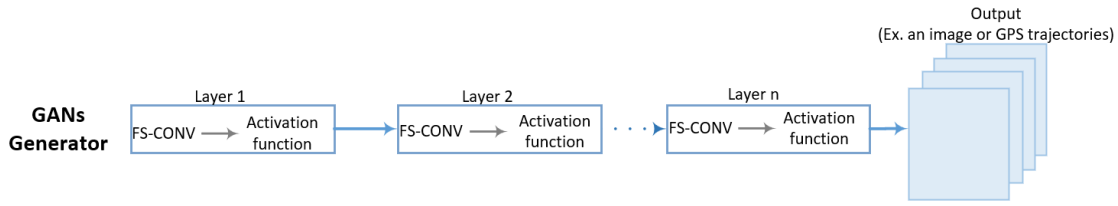
C.3.1 Mathematical Procedures and Architecture

The most well-known GANs architecture is the Deep Convolutional GAN (DCGAN) [157]. Most GANs use at least some of the architectural innovations proposed in the DCGAN architecture. As shown in FigureC.3c, in the DCGANs architecture, the discriminator consists of the convolutional layer(s) (explained in the next section) followed by a non-linear activation function. Afterward, there are one or two fully-connected layer(s). The final layer produces the class probabilities through a non-linear activation function such as softmax (for multi-class classification), sigmoid or tanh (for binary classification). As shown in FigureC.3b, the generator consists of several hidden layers, however, instead of convolutional layers, fractionally-strided (also referred to as transposed) convolutional layers are used. The final layer of the generator produces the desired output shape that can be an image or, in our case a series of GPS trajectories.

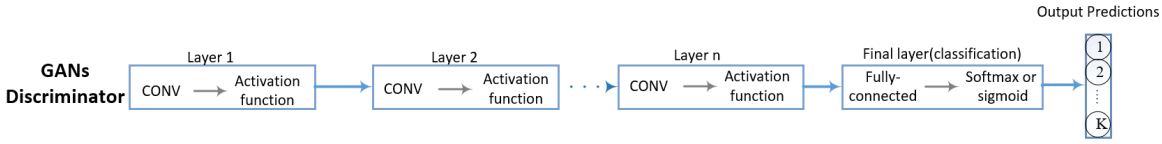
The building blocks of DCGANs are mainly borrowed from CNN architecture. As a result, the next section explains terms and concepts from the CNN literature, necessary to understand the architecture proposed in this paper. Also, since the results of our semi-supervised GANs model are compared with those of a CNN model, we explain the pooling operation most typically used in CNN models.



(a) General architecture of convolutional neural network (CNN)



(b) General architecture of GANs Generator.



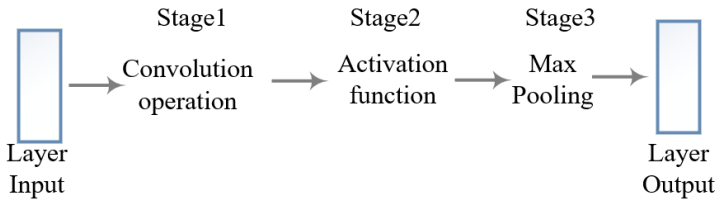
(c) General architecture of GANs Discriminator.

Figure C.3: General architecture of Different Models. (CONV: Convolution, FS-CONV: Fractionally-Strided Convolution

C.3.2 The Transposed Convolution Operation

AS mentioned previously, a typical layer of a convolutional neural network has three stages: the convolutional operation, the non-linear activation, and pooling. The convolution layer(s) transform(s) the input data into the desired output shape (class logits, or probabilities). These stages and how they are connected are shown in Figure C.4a We briefly describe each stage below. In a GANs generative model, we need a transformation going in the opposite direction, i.e. from something that has the shape of the output to something that has the shape of its input. To do this, transposed (fractionally-strided) convolution is used (further explained below). However, the pooling layer is not used in a transposed convolutional layer, as shown in Figure C.4b.

The generator model in the GANs framework needs to transform a latent variable input (such as variable z defined in Section 2.2.5) into an image, GPS trajectory or sound clip. This procedure requires mapping input to a higher-dimensional space [90]. For example, in Figure 2.5 the convolution operation maps from a 7-dimensional space to a 5-dimensional space, and by transposed convolution, we want to go the other way around. The transposed convolution operates by swapping



(a) Typical Mathematical Procedures Involved in a Convolutional Layer

(b) Typical Mathematical Procedures Involved in a Transposed Convolutional Layer

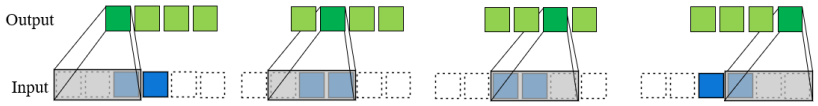
Figure C.4: Typical Operations in Convolutional and Transposed Convolutional Layers

the forward and backward passes of a convolution [90]. Hence, we can consider transposed convolution on input as the result of a direct convolution applied to an initial feature map [90]. Using this representation, assume we want to implement a transposed convolution on a 4×1 input (see the top row in green in Figure C.5a) with a kernel size of 3×1, unit stride and no zero padding. The transposed convolution here can be considered as a direct convolution on a $[2 \times 1]$ initial tensor with a $[3 \times 1]$ kernel and a $[2 \times 1]$ border of zeros with stride=1. The kernel starts from the left-most element of the bottom row and slides with stride=1 until it reaches the far right.

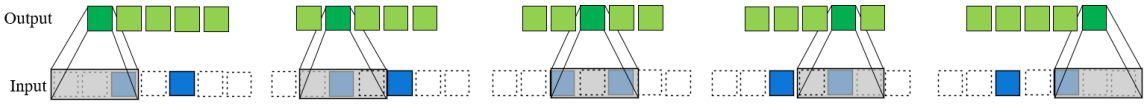
In Figure 2.4b the convolution operation can be represented by a sparse matrix C as below. The output in Figure 2.4b can be obtained by multiplying the matrix C with input tensor.

$$c = \begin{bmatrix} w & x & y & 0 & 0 & 0 & 0 \\ 0 & w & x & y & 0 & 0 & 0 \\ 0 & 0 & w & x & y & 0 & 0 \\ 0 & 0 & 0 & w & x & y & 0 \\ 0 & 0 & 0 & 0 & w & x & y \end{bmatrix}$$

Based on this representation of convolution operation, the backward pass of a convolution is simply



(a) The transpose of convolving a 3×1 kernel over a 4×1 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$). It is equivalent to convolving a 3×1 kernel over a 2×1 input padded with a 2×1 border of zeros using unit strides (i.e., $i^{'} = 2$, $k^{'} = k$, $s^{'} = 1$ and $p^{'} = 2$).



(b) The transpose of convolving a 3×1 kernel over a 5×1 input using 2×1 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 0$). It is equivalent to convolving a 3×1 kernel over a 2×1 input (with 1 zero inserted between inputs) padded with a 2×1 border of zeros using unit strides (i.e., $i = 2$, $i = 3$, $k = k$, $s = 1$ and $p = 2$).

Figure C.5: Two Examples of a Transposed Convolution Operation

derived by transposing matrix C. In other words we can take the output in Figure 2.6 and produce the input by multiplying it with c^T.

Another way of implementing the transposed convolution is to insert zeros between the input elements as shown in Figure C.5b. This causes the kernel to slide at a slower pace than with a unit stride and this is why in the literature sometimes the transposed convolution is called fractionally strided convolution [90].