NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

RODRIGO ROCHETEAU WAHNON FERREIRA

Bachelor in Electrical And Computer Engineering

VEHICULAR MOBILITY PREDICTION FOR IMPROVED NETWORK MANAGEMENT

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
2022, September

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

VEHICULAR MOBILITY PREDICTION FOR IMPROVED NETWORK MANAGEMENT

RODRIGO ROCHETEAU WAHNON FERREIRA

Bachelor in Electrical And Computer Engineering

Adviser: Nuno Miguel Abreu Luís
Adjunct Professor, Instituto Superior de Engenharia de Lisboa

Co-adviser: Rodolfo Alexandre Duarte Oliveira
Associate Professor with Habilitation, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING
SPECIALIZATION IN TELECOMMUNICATIONS

NOVA University Lisbon
2022, September

Vehicular Mobility Prediction For Improved Network Management

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.4) [25].

Acknowledgements

Firstly I would like to deeply appreciate the work done by my adviser, Miguel Luís. I offer my heartfelt recognition of your commitment to support me throughout our work, which wouldn't be possible to produce without you backing me up. You guided me when I was lost, and for that I give you my sincerest thank you.

To my co-advisor, Rodolfo Oliveira, I would like to thank you for your dedication, and more importantly, your patience. I know that sometimes it was though and arduous, but we made it this far. Therefore, I would like to congratulate you as well, this work is not mine but ours.

I would also like to thank the Instituto de Telecomunicações and the Aveiro Tech City Living Lab staff for the dataset generated by the vehicular network and the possibility to deploy and validate the prototype developed throughout this work. I would also like to thank project InfoCent-IoT (POCI-01-0145-FEDER-030433) for the conceptual background used in this dissertation.

I cannot go without thanking the institution that made my journey possible, NOVA School of Science and Technology, and all its personnel. I am now able to start my professional adventure, confident on my own capacities, knowing that in the past few years I gained a good foundational knowledge that will help me through now on.

A special thank you goes to all my friends and colleagues. During these years we studied, suffered and laughed together. Incredible bounds were made which I hope to take with me for the rest of my life. There are to many names to mention, but I am certain that they know who they are.

Another special thank you for all my family, to the ones who support me every day, and to the ones who are not here anymore. You have helped me since the first day and I know that you will continue to. To all, thank you for pushing me to be better, and especially for inspiring me.

Last but not least, at all, to my sweetheart. We have come a long way since the day we met each other, and you have encouraged me to test my limits since we were just high-school students. Happy to say that we did start this journey together, and together we will close it, now as engineers. Let us see what the future holds...

"He who contrives, defeats his purpose; and he who is grasping, loses. The sage does not contrive to win, and therefore is not defeated; he is not grasping, so does not loose." (Tao te Ching)

Abstract

The improvement of the communication networks' capacity has been a topic of continuous research in the last decade. We have been experiencing an increase in the amount of data demanded by users which anticipates that network requirements will increase substantially, even surpassing the legacy networks' capabilities. Therefore, it is imperative to conceive networks, or obtain improvements on the current ones. This dissertation proposes to use the mobility of non-stationary entities, such as vehicles, in order to improve the management of a telecommunications network, ultimately improving the resources' usage. With this improvement, it would be possible to observe a decrease in the amount of time required to support different services, such as the download of a content hosted in the network servers.

The main goal of my work is to implement a mobility prediction service. Reports with the current locations of mobile nodes are received, pre-processed and stored. Later on, once the required data is acquired, this service should use it to train a mobility prediction model based on machine learning techniques, capable of performing accurate predictions about the mobile entities' movements. The developed prototype should then be used in parallel with other applications that require information about these predictions.

Before implementing the prototype, multiple machine learning models were developed and validated to perform the mobility predictions. The study of these models was supported by also studying how different factors, such as the sequence length and spatial resolution, affect the predictions. Beyond this study, it was also researched how the mobility prediction would improve networks management, specifically by analysing the content's download time, from a end-user's perspective.

This prototype was developed and tested to be used in the introduction of a pre-caching mechanism on an existing network. However, it can also serve different services.

Keywords: Machine Learning, Mobility Prediction, Pre-Caching, Communication Network Performance Evaluation

Resumo

A otimização da capacidade de redes de telecomunicações tem sido um tópico de estudo constante durante a última década. Temos vindo a observar um aumento da quantidade de dados utilizados pelos utilizadores o que antecipa uma subida dos requisitos das redes, que irão ultrapassar as suas capacidades atuais. Por isso, é imperativo desenvolver novos tipos de rede, ou aumentar a capacidade das atuais. Esta dissertação propõe o uso da mobilidade de certos utilizadores, como veículos, numa tentativa de melhor a gestão de redes de telecomunicações, e assim otimizar a utilização dos recursos existentes na rede. Com esta melhoria, pretende-se diminuir o tempo necessário para prestar certos serviços, como é o caso do *download* de conteúdos.

O objetivo global do meu trabalho é a implementação de um serviço de previsão de mobilidade. Sucessivos relatórios com as localizações dos utilizadores móveis são recebidos, pré-processados e armazenados. Quando o número de dados adquiridos for suficiente, estes deverão ser utilizados para treinar um modelo de predição de mobilidade baseado em técnicas de *machine learning*, capaz de prever os movimentos dos utilizadores. O protótipo desenvolvido deverá então ser utilizado em paralelo com outras aplicações que necessitem da informação referente a estas previsões.

Previamente à implementação do protótipo em si, foram desenvolvidos e validados modelos de *machine learning* responsáveis pelas previsões de mobilidade. O estudo destes modelos foi ainda suportado pela avaliação de como diferentes fatores, nomeadamente o tamanho de sequência ou a resolução espacial, afetam as previsões. Para além disto, também foi investigado como é que a previsão de mobilidade melhoraria a gestão das redes de telecomunicações, analisando especificamente o tempo necessário para fazer *download* de determinados dados, do ponto de vista do utilizador final.

Este protótipo foi desenvolvido e testado visando a implementação de um sistema de *pre-caching* numa rede de já existente. Contudo, pode também funcionar com outros serviços.

Palavras-chave: *Machine Learning*, Previsão de Mobilidade, *Pre-Caching*, Avaliação de Performance de uma Rede de Telecomunicações

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AN Attention Network 24, 77
ANN Artificial Neural Network xvi, xvii, 3, 15, 16, 33–35, 37, 39–42, 49, 50, 57, 76, 77
ATCLL Aveiro Tech City Living Lab xvi, 3, 45, 53, 62

BGP Border Gateway Protocol 9
BN Bayesian Networks 14
BP Bundle Protocol 9
BS Base Station 27–30

CAM Cooperative Awareness Messages 45
CDF Cumulative Distribution Function xvi, 37, 38, 48, 49, 53–55
CDN Content Distribution Network 51, 53, 56
CNN Convolutional Neural Network 20

DBN Dynamic Bayesian Network 14
DTN Delay Tolerant Network 8

eHMP enhanced Handover Mechanism using Mobility Prediction 28

FNN Feedforward Neural Networks 20

GPS Global Positioning System 26, 27, 30

HMM Hidden Markov Model xv, 12–14, 27, 28

ICN Information-Centric Networks 28
IoT Internet of Things 1, 62

LSTM Long Short-Term Memory Network xv–xvii, 3, 22, 23, 25, 33–35, 37, 39–42, 49, 50, 57, 64, 66, 76, 77

MAC Media Access Control 61
MC Markov Chain 12–14, 26–29
MPL Multilayer Perceptron 20
MQTT Message Queuing Telemetry Transport 62, 70
MSE Mean Squared Error 17, 34, 50, 66

OBU On-Board Units 45, 53, 61, 62, 76, 77
OSPF Open Shortest Path First 9

PET Path Expiration Time 26, 30
PLB Proactive Load Balancing 27
PP Prediction Performance xvi, 36, 43, 50, 51, 53, 55, 57, 67, 69, 76

QoE Quality of Experience 27, 28, 30
QoS Quality of Service 26, 27, 29

ReLU Rectified Linear Unit 34, 49
RNN Recurrent Neural Network 21, 22, 25, 26
RSU Road Side Unit xvi, 27, 28, 30, 43, 45, 46, 49, 52–54, 58–62, 66, 70, 76
RTT Round Trip Time xvi, 43, 53–55

SGD Stochastic Gradient Descent 19
SONs Self-Organizing Networks 27

TCP Transmission Control Protocol 8, 61

UDP User Datagram Protocol 8

VLPHA Vehicular Location Prediction Handover Algorithm 29

1

Introduction

Over the last decades, humankind has seen incredible innovation and development in numerous technology fields, bringing more possibilities and aspirations to the scientific community. Technology is now more present in our daily lives than ever before, and this trend shows no indications of stopping. Modern societies are the greatest example of it, where this is demonstrated by changing the way we make business, or by the creation of numerous devices that aim at making our lives easier. One irrefutable example is the technologies employed during the recently lived pandemic era, which enabled many to continue their professional and personal life in the most convenient way possible at the time. The communications field is no stranger to this fast-paced development as well and manifested itself as a central pillar in our society. It has also been demonstrating incredible progress not only in terms of connectivity but also in transmission rates, while not dismissing from mind the security issues that have been becoming more and more important to handle.

These developments made it possible to implement concepts that before were nothing more than utopias, as is the case of hot topics such as 5G systems, Internet of Things (IoT) in conjunction with the entry into the Industry 4.0 realm, or even networks where the data is stored in a decentralized manner, such as blockchain networks. All of them rely on a concept where numerous devices are securely connected, creating a network where information can be acquired, stored, and ultimately processed, allowing to emerge a lot of different potential applications. However, these technologies require some crucial factors to be assured at a communication network level such as available bandwidth, availability, reachability, and low delay.

On a different note, the nature of the information being consumed by the average user is also changing. For one, the type of content that we, as users, access on a daily basis is becoming more and more complex. Nowadays, we use communication networks for example to interconnect console players throughout the world in real-time online games or to stream videos in higher resolutions than before, which are very demanding. And on the other side, the amount of data being consumed nowadays has increased to an unprecedented level.

1.1 Motivation

Due to the innovation that is taking place in terms of the end devices and offered services, the current network infrastructures and protocols should be improved in terms of data processing and routing capabilities or will, otherwise, suffer immensely from this increase in data. Not only the bandwidth requirements will become more harsh, but the actual processing performed at the network device level will also start to be heavier due to the number of data flows the devices will need to handle.

To address this issue it is possible to take advantage of non-stationary end devices' mobility. One proposed idea by the academic community is the creation of an alternative channel, or in other words, an alternative path for the information to flow from one point to another by using the devices' movements. Instead of utilizing only the resources available in the typical communications infrastructures, we can take advantage of mobility by equipping these mobile entities with the appropriate resources such as communications or storage devices. Another possibility is to use the knowledge about the mobile entities' mobility, not to construct a substitute network, but to enhance the performance of the current communications networks that are supported by the existent infrastructures. By using the knowledge about the entities' mobility we can implement a variety of services that have the ultimate goal of improving the management of communications network management.

When trying to employ these concepts, machine learning may act as a powerful tool to put together more robust and reliable networks, by determining the entities' future movements. Knowing their trajectories would allow us to know, for example, if a determined mobile entity would be appropriate to transport some information to a certain place. On the other hand, by only predicting their mobility, it is also possible to build services that will improve resource management supported by the information achieved by mobility prediction algorithms based on machine learning.

1.2 Objectives

The ultimate objective of our work is the development and implementation of a mobility prediction service, intended to cooperate with other applications. The services offered by these applications can be very different, but always have the intention of taking advantage of accurate mobility predictions to improve the performance of the communications network management. However, the mobility prediction service implementation has to be preceded by other tasks. Firstly, it is required to survey the existing work on machine learning, and perform the implementation and validation of prediction models in an offline manner. Multiple prediction models are to be developed throughout our work, while using different machine learning approaches in different scenarios, in order to verify our capacity to accurately perform mobility predictions.

Another key aspect is to verify the enhancements that these mobility predictions can

bring to a communications network. To achieve this, we assume that our prototype is going to serve a pre-caching mechanism and verify the enhancements brought in terms of data's retrieval time from the end-user point of view.

1.3 Contributions

The contributions of this work are listed as follows:

- Development of various machine learning models to perform mobility predictions, focusing specifically on Long Short-Term Memory Network (LSTM) and Artificial Neural Network (ANN) approaches. Besides analysing each model's performance, we also compare both approaches in terms of computation time.
- Study how sequence length and spatial resolution influence the mobility prediction algorithms. This study is supported by a conference paper published during the development of the dissertation's work, specifically: R. Ferreira, M. Luís and R. Oliveira, "The Impact of the Spatial Sampling Resolution on the Prediction of Vehicular Mobility", 2022 International Wireless Communications and Mobile Computing (IWCMC), 2022, pp.425-430, doi:10.1109/IWCMC55113.2022.9824986.
- Verification of the benefits of a pre-caching mechanism implementation, supported by a mobility prediction model. This study takes into account the data's retrieval time when a user intends to download a specific content present in the Cloud.
- Implementation of a mobility prediction prototype that receives information in real-time, concerning the location of some mobile entities (buses) travelling in Aveiro's city, which is used to develop a mobility prediction application that performs predictions as requested by a third-party service. This prototype is deployed in the Aveiro Tech City Living Lab (ATCLL) and available to be used by other applications.

1.4 Document Layout

This report is organized as follows. Chapter 2 presents an overview of previous works. Firstly, we focus on the importance of deploying new forms of communication channels and why the use of mobile entities is a viable option. Then, multiple different learning algorithms are described while differentiating machine and deep learning. Lastly, we present multiple studies that exhibit how predicting mobile entities' mobility may bring enhancements to communications networks management.

In Chapter 3 we approach the mobility prediction problem. In a first stage, we present the methodology used to construct two machine learning models that employ different approaches, as well as their validation in a generic way. Then, we perform a characterization of the dataset used, which concerned multiple taxis travelling through Porto's city. To finalize, we characterize the models' prediction performance.

In Chapter 4, our goal is to show that implementing a pre-caching mechanism could be beneficial to a communications network. In order to obtain concrete results, we decided to take two specific scenarios, one in Porto and another in Aveiro, and study how the networks would respond to this implementation. Firstly, we describe both scenarios and perform the characterization of both datasets. Then we evaluate the prediction accuracy achieved in both cases. Finally, we identify the benefits gained by using the prediction models to implement a pre-caching mechanism.

Chapter 5 is dedicated to the development of the prototype. It starts with a description of the structure of the prototype itself. Then, we explain the infrastructure and methodology used, that could also be employed to develop any similar prototype. Afterwards, we describe the creation and validation of the prototype's prediction capacities. The last section includes final thoughts about the developed prototype, as well as the presentation of parts of the implemented code, and the showcase of the application functioning.

To conclude the dissertation, in Chapter 6, we perform an overview of the work carried out, and of some final thoughts. Then, we present some ideas for future work.

2

Related Work

2.1 Importance of Mobility in Future Networks

It is well known and documented that every technological field is in faster growth than ever before, and telecommunication networks are no exception. There are numerous proves to this statement, such as the increase of data transmission rates or the service quality possible to be obtained in applications such as video streaming. According to Cisco, in terms of predictions for the year 2023, it will be possible to see a large increase in the number of internet users reaching somewhere around two-thirds of the population (5.3 billion users), an average of 3.6 devices with access to IP networks per capita which translates to a total of 29.3 billion devices and speeds that will reach 110.4 Mbps for fixed broadband networks and about 575 Mbps for 5G networks [3].

The fact we are seeing a technological revolution in terms of access to connectivity, and dealing with increasing internet traffic, as mentioned before, despite demonstrating the great improvements done in the field, also raises problems. At this developing rate, if there is no innovation in new techniques to replace or improve the traditional networks' performance, or enhancements in the infrastructures, the networks will be overwhelmed in the near future because the end devices are being the focus of ongoing innovation, therefore more connections and higher quality is required. In other words, if none of these measures is taken into place, the core of the networks will have insufficient capacity to deal with the demanding flows of data throughout the world. However, investing in the infrastructures is not sufficient since it represents an expensive investment while only postponing the problem.

Aside from that, another problem that the current networks, also known as legacy networks, present is the so-called digital divide. This refers to the lack of connection coverability in some parts of the globe, namely in developing countries, secluded rural areas with low population density, and conflict zones. This may appear, in some cases, as an easy to fix issue, however, since the investments required to deploy the legacy networks and its infrastructures are so high, it becomes unpractical to do so.

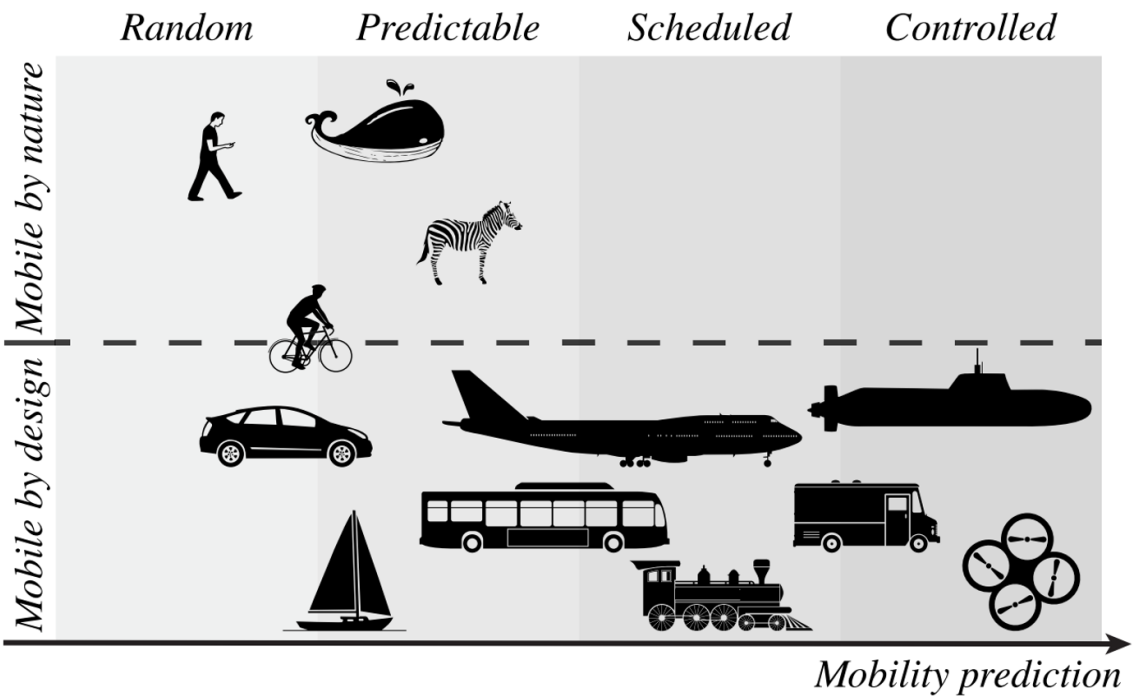


Figure 2.1: Possible classifications of mobile entities based on their movement essence and on their movement predictability, taken from [5].

In light of these two problems, a possible solution that has been studied and documented in the past years is utilizing mobility to substitute the traditional networks. Specifically, this consists of using mobile entities to establish a Sneakernet, which refers to a network where data transmission relies on physical movements, such as someone storing a photograph in a USB drive and actively walking to another's computer to insert it and transfer the photograph. In this type of network, the mobile entities are the ones moving, and they are equipped not only with a storage device but also with some means of communicating with other nodes such as a wireless interface [5]. The epicentre of this approach is obviously the mobile entities, and these can be animals if equipped with the required devices mentioned above, humans carrying digital devices, cars, planes, and even drones. Furthermore, the mobile entities can be classified in mobile by nature or by design, and also according to their movement predictability as random, predictable, scheduled, or controlled. A representation of these classifications can be found in Figure 2.1.

By taking advantage of the mobile entities, we can then implement this Sneakernet to create a communication channel that relies on their movement. In this case, the data is not only transmitted from the origin to the destination through the use of legacy networks, but also by these entities. By using the mobility to establish a communication channel, it is then possible to tackle both problems raised above:

- Regarding the capacity problem of the networks, it is possible to deviate the delay

tolerant data making it go through this alternative channel. As an example, we can consider weekly backup files from a company that represents large data amounts being transferred to its home office in another city or even another country. Instead of overloading the legacy network, this transfer can utilize a train or a plane travelling between the two cities to take the data at least closer to its destination. This will result in an increase of the bandwidth available by introducing a channel provided by the entities' movement.

- When considering the lack of coverage in a distant villa, for example, it is possible to use the cars and the buses travelling in its direction to transport data from a gateway in the nearest city until it reaches a sink point in the villa, where all the data will be gathered and further processed. The implementation of this network will actively bring connectivity to a location where there was none before, therefore, in this case, replacing legacy networks.

It is important to note that various possible configurations exist for this type of network, where some are more suitable to certain cases than others. There are some aspects to consider when choosing which schemes to utilize, being the main ones are the type of issue that this deployment will handle and the type of entities' mobility. Having more knowledge of their movements will typically result in higher delivery rates while requiring less packet replication. Since it is more efficient and challenging, The main focus of this work will on predictable entities since they are more relevant and challenging that the scheduled or controlled ones.

As said above, having in mind the goal of the channel deployment, numerous variants have to be taken into account when designing it, such as the types of entities used, how the entities communicate with themselves, among many others. However, the most important aspect may be what type of design is used. Assuming that our goal is to deploy a new communication channel to connect origin location A to destination B, there are multiple possibilities to achieve it. The most straightforward option is the construction of an infrastructure between them, to connect them via legacy networks which may represent a considerable investment. If this is not the most desirable option, it is possible to use mobile entities, but it requires choosing a type of design as well. The following paragraphs will mention some types of designs, which can also be found in Figure 2.2.

It is possible to achieve this by direct data delivery where the entity that picks up the data at A is the one delivering it to B, therefore depending on only one entity, or a single group of entities, to carry the data. The direct data delivery approach can be further categorized into a passive method where the entity's movement is taken advantage of, the active method that consists of somewhat controlling and modifying the entity's movement, or even into the paying method that relies on paying for a service that will transport the data, usually by post or delivery companies. The performance, especially in the passive method, will be affected by the entity's movement predictability, where predictable or scheduled ones may result in a better delivery ratio.

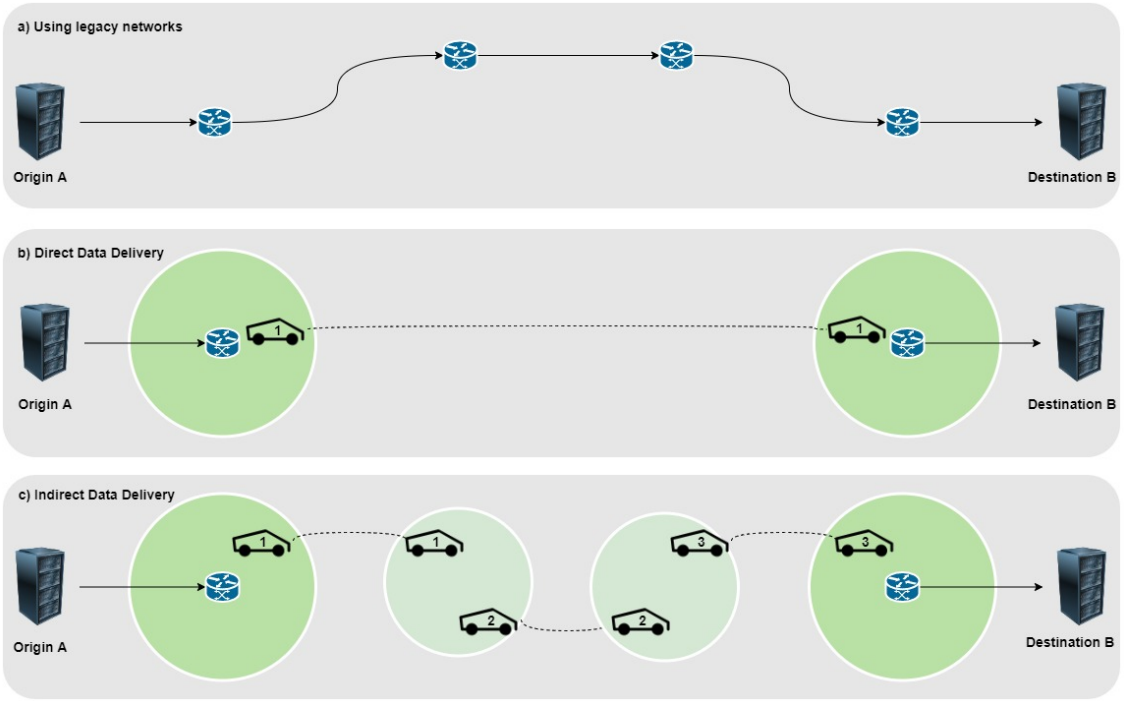


Figure 2.2: Simplified diagrams of different approaches, such as a) deployment of conventional infrastructures for legacy networks, b) use direct data delivery approach, or c) the indirect data delivery, in this case using the synchronous method.

Another possibility is to use indirect data delivery that, by opposition to the first, will depend on multiple entities such as relay nodes [12]. The indirect data delivery approach may also be broken down into different methods as synchronous or asynchronous. On the one hand, in the synchronous method, it is required that the different entities are in contact with each other to pass the data from one to another, either at any pre-specified location which refers to the pre-positioned composition or at any location the two entities meet referring to the floating composition. On the other hand, the asynchronous method consists of data transmission when the entities do not need to encounter. This is achieved by using intermediate nodes that receive the data from one entity, buffer it, and then pass it to the other mobile entity when in its proximity. The intermediate nodes may be stationary appropriate for when the routes of the entities intersect at a specific location but not at the same time or can be intermediate mobile nodes for when it is required to bridge the routes of the entities when they do not intersect [5].

These networks are also known as Delay Tolerant Network (DTN) since they are appropriate for scenarios such as the ones described in section 2.1, mainly due to their capacity to overcome problems like delay, considerable error rates, and intermittent connectivity. Subsequently, the transmission protocols usually adopted in legacy networks, such as the Transmission Control Protocol (TCP), or User Datagram Protocol (UDP) do not correspond to the expectations here. In DTNs, a transmission protocol widely used

is the Bundle Protocol (BP) that forms a network by having the entities buffering, carrying, and then forwarding the message to another entity, rather than implementing an end-to-end delivery, and does not consider that connectivity between two entities is ever ensured [32]. When it comes to routing protocols, the ones typically used, such as Border Gateway Protocol (BGP) or Open Shortest Path First (OSPF), are not well fitted for these scenarios as well. There are numerous possibilities on how to configure and manage these networks, especially when having some sort of control or knowledge on the entities' mobility.

The following section will survey possible learning algorithms to use in this mobility prediction, while section 2.3 will analyze different possible improvements on network management made with this mobility prediction.

2.2 Machine and Deep Learning tools for prediction

2.2.1 Machine Learning

Over the last few decades, humanity has seen a tremendous increase in the rate at which the technological world advances. It is undeniable that new technologies are continuously emerging and evolving, and it is difficult for us to stay up with them. The fast progress brought new challenges to the scientific community, particularly in the computer science field, which expanded perhaps like no other area during this period. Machine learning is a topic that has attracted a great level of attention and nowadays is growing at an increasing pace, even though previously we had seen it stalled for some time. Accordingly to [26], this stagnation and posterior development is simply explained by the lack of easy access to storage and computational resources at the time.

Nevertheless, after all, what is machine learning exactly? Machine learning is a subfield inside Artificial Intelligence which, as the name suggests, refers to the process by which a machine is able to learn something, specifically by being supplied with a large quantity of data. In this way, the machine will analyze the data we feed it and begin the learning process by identifying patterns. This will enable the computer to perform a task-oriented role without being specifically developed to do it, thus classifying images and making predictions on probabilities of certain events, among others. These may seem vague and pointless abilities. However, these are the same capabilities that allow us to use language translation, to unlock our smartphones and computers by facial recognition, or to filter all the spam on our emails which would be otherwise impractical, almost impossible to use. Machine learning is suitable to be used in such a wide range of applications due to its characteristics. However, it is important to note that the term machine learning does not refer to an all-problem solver, it involves multiple types of algorithms, each of which is suitable for use in specific scenarios. Besides that, we as developers always feed the algorithms with data, known as data training set, which, depending on different types of data sets possible, will categorize the algorithm into three categories [6]:

- **Supervised Learning –** when classified as a supervised learning algorithm, it means that the data set has two distinct pieces of information: the input features and values that allow the algorithm to make its prediction and the expected output. Thus, in practical scenarios, the algorithm learns the associations between the possible outputs and all of the inputs. After the prediction calculations, it is computed the deviation of itself from the expected value, and this is what makes it possible to quantify how satisfactory the output is, a process which is analysed further in-depth later. As an example of supervised learning, we can think of a scenario where the algorithm role is to determine if a particular object is a tablet or a mobile phone-based on input features such as weight and size. The algorithm will learn with the data provided both on tablets and phones and afterwards make predictions, hopefully correct, when provided with an unknown sample.

- **Unsupervised Learning –** unlike the previous case, in unsupervised learning the data training set only contains the input information. Consequently, since there is no information about an expected value, in this category we do not find any function that controls the deviation so, what the algorithm is doing, is trying to detect patterns in the data so that is possible to create associations between the different input features. Unsupervised learning is often used in clustering problems, for example, when a company's marketing team needs to classify customers in groups to advertise more relevant publicity to them. An algorithm such as this would be able to cluster people into different groups based on different features like gender, age and types of previous purchases, among others.

- **Reinforcement Learning** - this method is optimal for scenarios in which the environment is not deterministic. Since it is susceptible to changes, it is not possible to calculate an exact error measurement. In reinforcement learning, the learning process depends on feedback from the environment itself, which is known as a reward, that classifies the model's action qualitatively. When the model performs a certain task, it will receive a reward that may be positive or negative and will adapt accordingly to the received rewards. The goal is to learn the policy, which is a set of actions to be taken, that theoretically will result in the highest number of positive rewards possible.

2.2.1.1 Training

Training a model is nothing more than analysing the output at each iteration and then tweaking, or tuning, some parameters based on how good or not this output is, so that the performance of the model improves. A great definition for this process was given by Herbert A. Simon in [29], which defines it simply as the adjustments made to the system that allows it to perform better the next time that executes a certain task, in the same environment. This process is analysed in depth later, but for now, is important

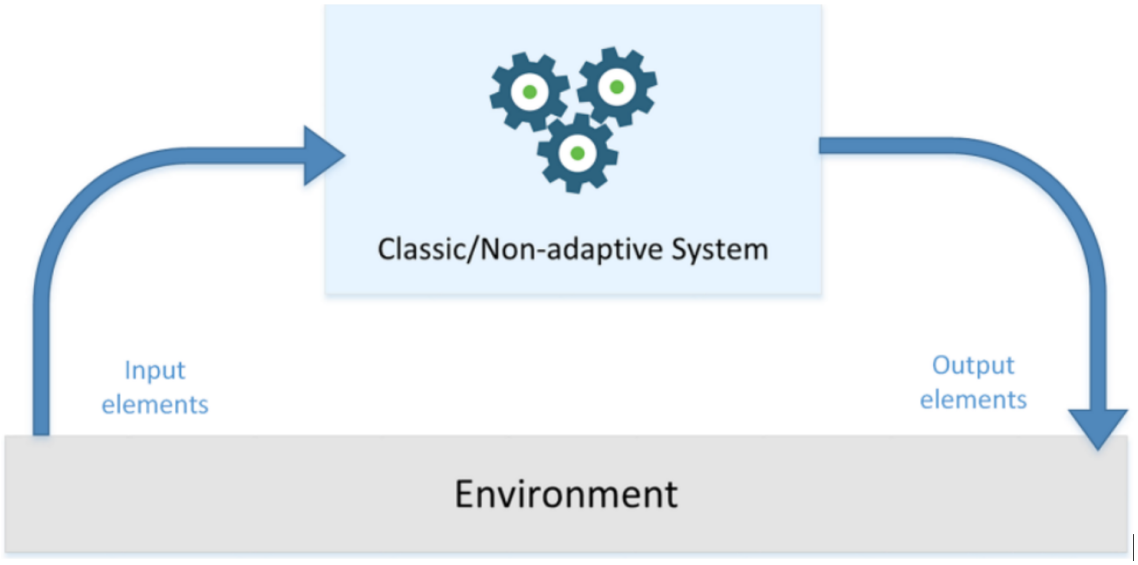


Figure 2.3: Representation of the interaction between the model and the environment when an non-adaptive system is considered, taken from [6].

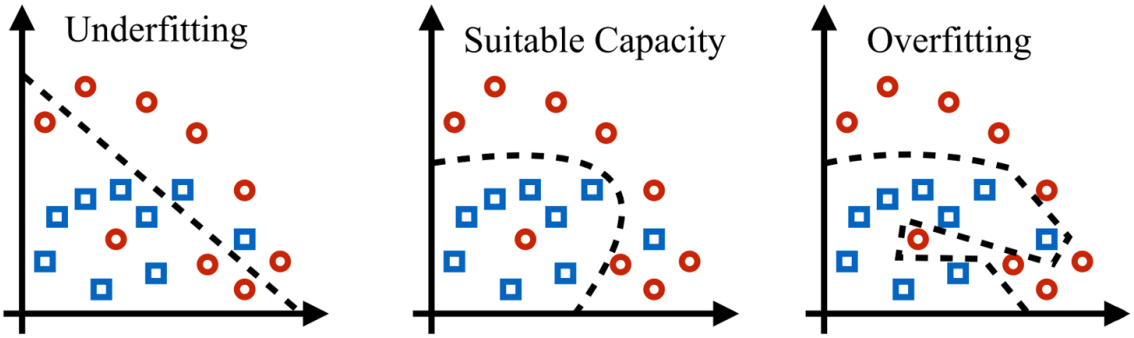


Figure 2.4: Representation of the three possible classifications of the model in terms of generalization, taken from [42].

to understand its concept. The Figure 2.3 refers to the interactions between the model and the environment. However, with this tuning a problem emerges: how generalized or specific should the model be? This is not straight-forward question, which could lead to two unwanted situations if not determined correctly.

In Figure 2.4 we illustrate three different model behaviours. The underfitting relates to the models that could not capture the dynamics in the data provided to it, leaving us with too many generalized scenarios. Overfitting is the opposite situation, relating to models that have such capacity, that are able to describe too much of the data set they were fed with, not demonstrating good performance when presented with different inputs. Therefore, succinctly, the aim is to generate and train a model so that it presents the lowest prediction error possible while still maintaining a reasonable amount of generalization.

To understand machine learning, it is essential to, as mentioned in [42], acknowledge that there are two distinct moments when we are talking about machine learning that

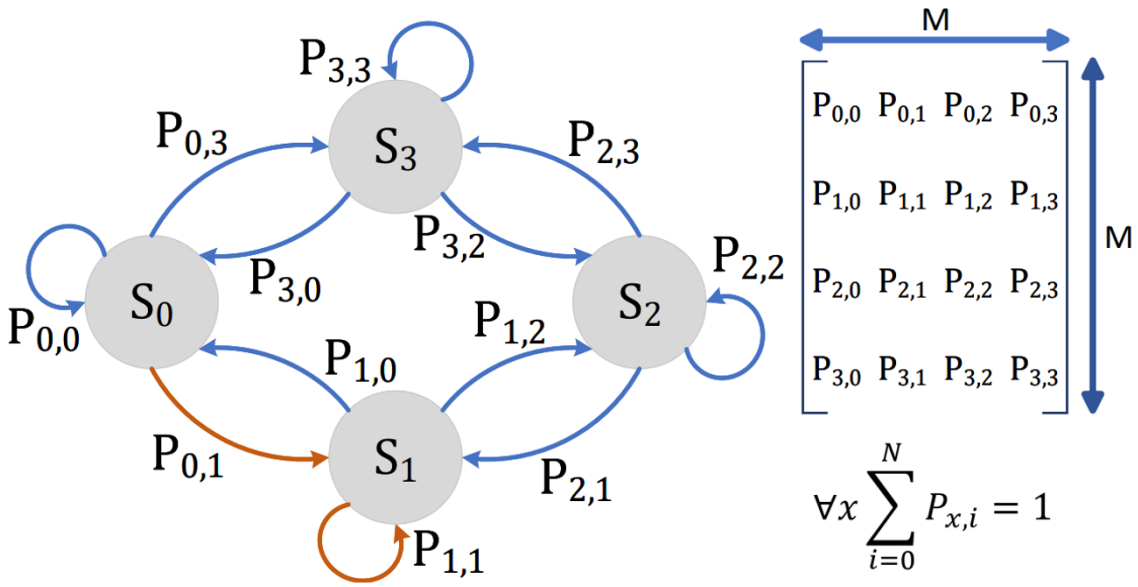


Figure 2.5: representation of a Markov Chain which consists of four possible states, M=4, and representation of the associated transition probability matrix, with M*M dimensions. Taken from [37].

should not be conflated, yet often are. To begin, there is the learning phase, also known as the training phase, in which the model is created and relies on the data provided to adapt itself and its parameters in order to be able to output satisfactory results. Afterwards, when the model is ready, there is the phase where the machine actually performs the task at hand, resulting in actual outputs, such as predictions.

2.2.1.2 Markov based methods

Markov based methods, especially the Hidden Markov Model (HMM) is a viable alternative to the more complex deep learning approaches in mobility prediction scenarios. HMM is worth mentioning since it has also been widely used in such problems in recent years because of its respectable results in scenarios with limited complexity. However, to comprehend the HMM it is necessary first to understand Markov Chains.

In a nutshell, Markov Chain (MC) is a model where the prediction of the future entity's location depends exclusively on the current one, thus being a memoryless model. It consists of the set of possible states and transitions between them, where the agent has a probability of remaining in one state and a set of probabilities of changing to another state at any given moment, also known as transition probability. As previously stated, it is a memoryless scenario, meaning that the most likely event is determined only by the current state. The model will discover patterns during the training phase and utilize them to make accurate predictions later. What training entails will be discussed in greater detail more ahead.

The important thing to understand is that MC is a simple model to implement, but

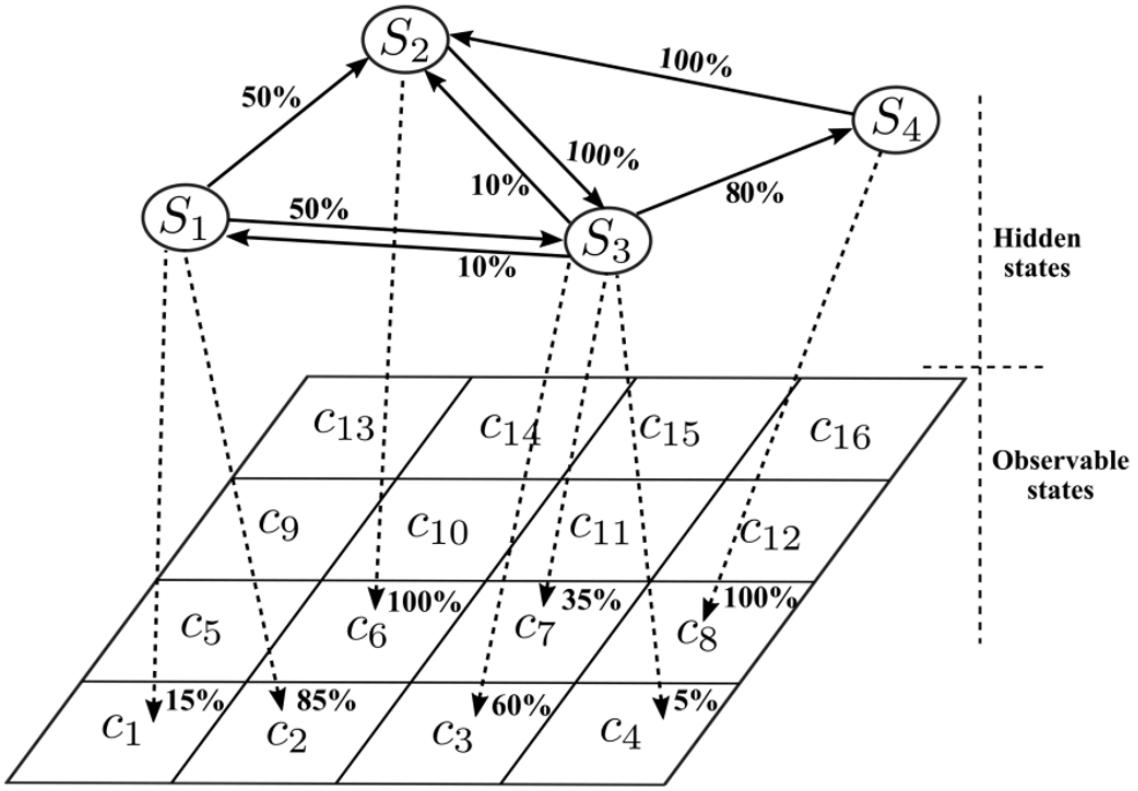


Figure 2.6: Example of a HMM in a mobility prediction scenario, emphasizing its observable states, cells, and hidden states, sets of consecutive cells. Taken from [18].

it does come with a cost. Its performance is strongly dependent on the computational time required to compute the transition probability matrix, which implies that, as the scenarios get more complex, more states appear and more transition probabilities are required to be computed, which will quickly degrade its performance and usability [44].

Later on, the HMM was created to complement the MC's principles, and it has been shown to produce superior performance results in most circumstances than its predecessor. In the HMM, it is introduced the notion of two different states: similarly, to the MC we have observable states which, as the name suggests, are states that we can visualize, but HMM also introduces the use of another stochastic process that cannot be observable, creating another set of possible states denominated hidden states. In mobility prediction scenarios where a map grid is considered, the observable states, are the visited cells and the hidden states, reflect every possible small sequence that may characterize the user's movement along the grid. This method can fit in the supervised learning as well as the unsupervised, depending on how it is implemented. This is shown in the Figure 2.6, where a grid with sixteen cells is considered, and where the user's movement may be described by four different sequences of visited cells.

Before defining an HMM model, a few parameters must initially be defined as stated in [18]:

- Sets of possible observable states of the model, represented by $O = \{o_1, o_2, ..., o_N\}$ where N is the total number of observable states.

- Set of hidden states, $H = \{h_1, h_2, ..., h_\Psi\}$, with Ψ being the number of unique sequences considered in the training phase.

- The HMM inference model, λ, with $\lambda = \{\pi, A, B\}$, where:

  - $\pi = \{\pi_1, \pi_2, ..., \pi_\Psi\}$ is the initial state distribution of the model.
  - $A = \{a_{1,1}, a_{1,2}, ..., a_{1,\Psi}, a_{2,\Psi}, ...a_{\Psi,\Psi}\}$ represents the state transition probability matrix.
  - $B = \{b_1(o_1), ..., b_1(o_N), ..., b_\Psi(o_1), ..., b_\Psi(o_N)\}$ is the emission probability matrix, also known as confusion matrix.

After defining the HMM model, the known data should be used to train the model, and thereby be able to, by using a line of observable states, determine which hidden state is more likely to describe the user's motion and therefore predict the next cell.

2.2.1.3 Bayesian Networks

A Bayesian Networks (BN) is a supervised learning method that consists of an acyclic graphic, which is characterized by G = (V,E), where the V and the E correspond, respectively, to the vertices and the edges of this graphic. These vertices represent every random variable taken into account in the model and come accompanied by a probability distribution table that demonstrates how the probability of the vertices are affected by their parent vertices, therefore being a probabilistic model, although this may not be so evident [9]. In Figure 2.7 is possible to see an example of what a basic BN looks like, this one being composed by 7 vertices and 17 edges.

When considering a mobility prediction scenario, using BN enables us to determine the most probable trajectory of the subject by using the attributes retrieved from the data provided to the model using Bayesian inference. With this type of model, by opposition with other alternatives like the MC, it is possible to take into consideration various factors that should affect the predictions, such as road topology or weather [18].

On a different note, there is a type of BN's called Dynamic Bayesian Network (DBN), which are networks where the vertices are sequences of state variables. This may seem familiar, as was previously said that the hidden states in the HMM, particularly in the mobility prediction scenarios, are sequences of visited locations. In other words, the HMM described before may be considered a DBN, and therefore also a BN, where the graphic vertices are the sequences that may describe the user's movement and the edges represent the transition between hidden states.

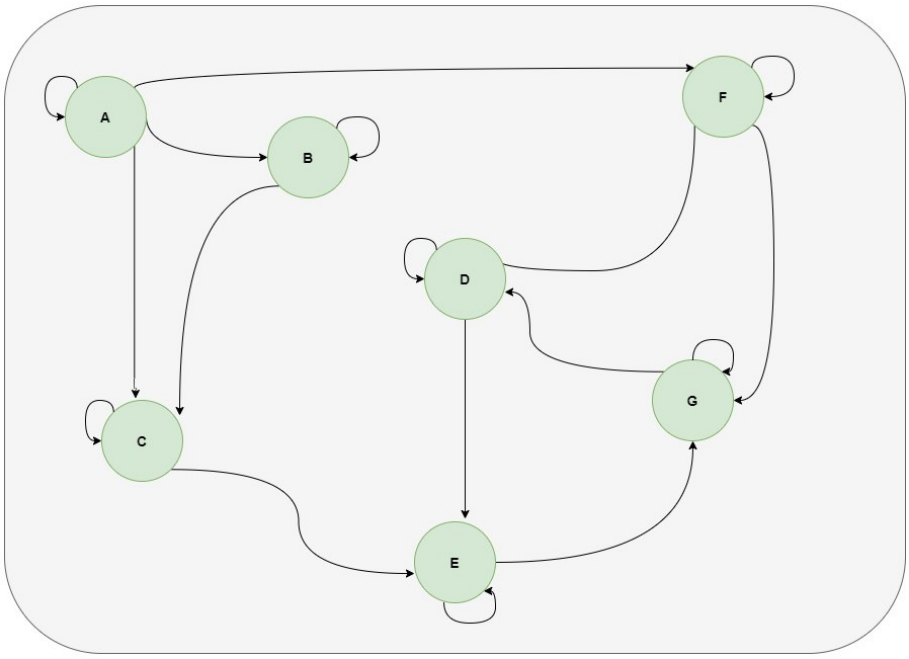


Figure 2.7: Simple example of a generic Bayesian Network with its vertices and its edges.

2.2.2 Deep Learning

One very trendy topic when it comes to machine learning is deep learning, also known as hierarchical learning. Deep learning has received a great deal of focus in recent years since it is more capable of dealing with complex problems in general, demonstrating better results, especially as the quantity of data available for training increases. It has also been shown to be excellent for extracting unknown characteristics in complex situations and is thus frequently utilized in speech recognition, text processing, and image analysis, among other applications.

Nevertheless, why is the deep learning approach different structurally? When discussing more classic machine learning algorithms, such as the previously mentioned Bayesian Networks, it is being considered structures with a relatively simple structure, often comprised of only one input and one output layer, [24]. On the other hand, deep learning adds the concept of depth, in which the model is made up of many layers. Each layer must conduct its own set of linear or nonlinear transformations; as a result, each layer gets its input, does the required mathematical calculations, and then delivers its output as the next layer input. This creates a very complex hierarchical network composed of various layers that try to approximate the complex problem to a set of simple operations, also known as artificial neural networks. The usage of neural networks, according to [36], is the centrepiece behind the deep learning success and potential.

2.2.2.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is a category of deep learning algorithms that have enabled great innovations in the field as well, and will be one of the algorithms in

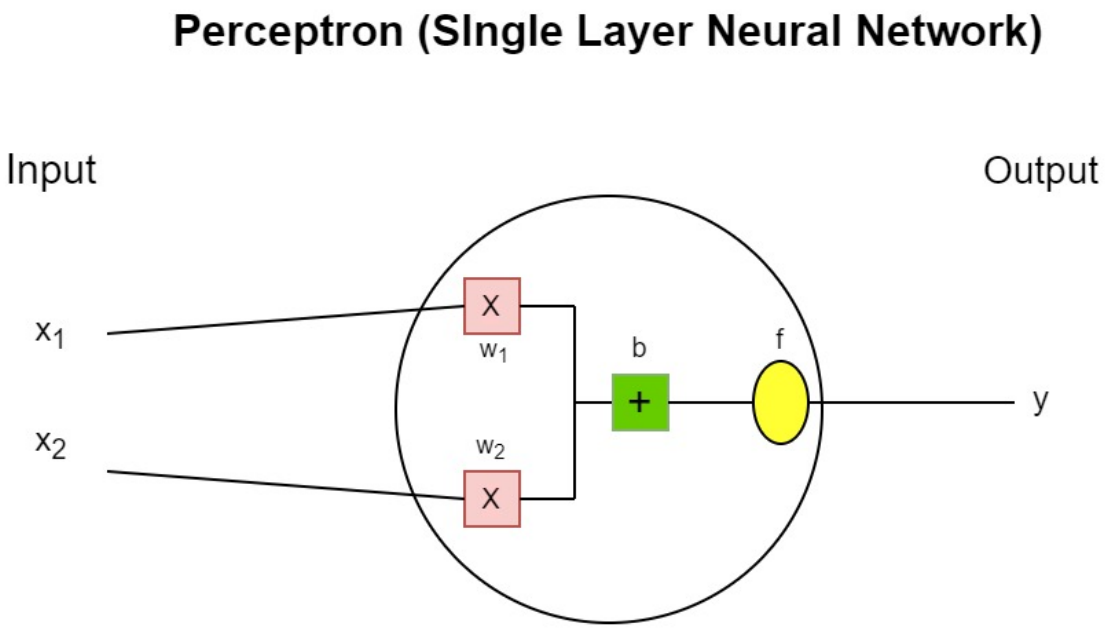


Figure 2.8: Graphic representation of a neuron, adapted from [26]

focus along this dissertation. Depending on its scheme and on the data provided to the model it can be used as a supervised or unsupervised learning method, however, its more common to be found in a supervised way. This last statement is also true for the rest of the models presented since they are variations of ANNs. As J. Loy states in his book, [26], ANN's importance proliferation is owed not only to its scalability but also its capacity of approximating and modelling any mathematical problem, regardless of its degree of complexity.

To be capable of understanding the foundations of ANN's, one must first comprehend its most fundamental unit, the neuron.

In Figure 2.8, it is possible to observe a generic illustration of what a two-input neuron which, in this case, may also be denominated as perceptron since it is a single layer neural network. Firstly, we have both neuron inputs x_1 and x_2 which are multiplied by the corresponding weight term w_1 or w_2. Afterwards, the results of both multiplications are summed up together in another term, the bias b. In the end, the output y is calculated by passing the sum value by an activation function, f, which limits the output to a certain interval. One widely used activation function is the sigmoid, limiting the output to values between [0,1]. All this procedure is translated into:

$$y = f(x_1 * w_1 + x_2 * w_2 + b), \tag{2.1}$$

where the loss function sigmoid, generically is given by

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.2}$$

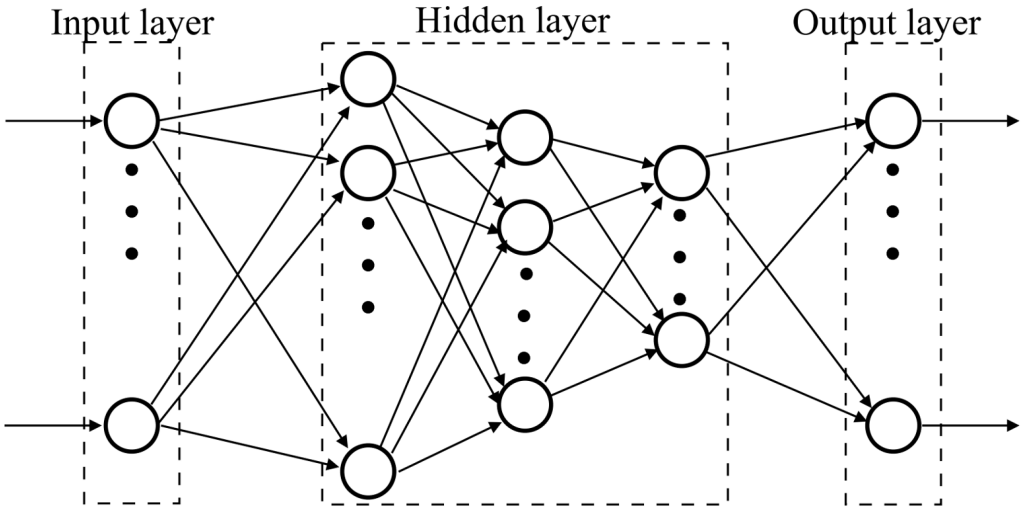


Figure 2.9: Basic scheme of a deep artificial neural network, taken from [42].

Once grasped the fundamental idea of a perceptron, it is easy enough to understand a simple neural network, which is just a collection of neurons linked together to form a network. Three different components can be identified: the input layer, which feeds the network with the data provided to the model, one or more hidden layers, which are represented by the neurons themselves and perform the majority of the processing, and finally the output layer, which may require some final and straightforward processing before outputting the prediction values.

Having understood the constitution of a neural network, an obvious problem is presented: what are the optimal values for the weight and the bias terms? Although this is not a straightforward question, the ultimate goal of network training is to identify such values for each neuron. However, in order to achieve the optimal values, it is necessary first to utilize a loss function, which enables the network to quantify the difference between the model output and the intended one, thereby indicating if the output is satisfactory or not. The Mean Squared Error (MSE) is one of the most common types of loss functions, and one of the used in this work, because of its comprehensibility, even though it is not optimum for rapid convergence. The MSE value is calculated as follows

$$L = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2, \tag{2.3}$$

where y_{true} is as denoted before the network output, y_{pred} represents the desired output and n is the number of data samples used in the calculation. The better the network terms are trained, the more similar both outputs are, the lower the loss function value is, and vice-versa. Thus, one of our goals while training a network, although not the only one, is to minimize this value since a low loss function is intrinsically correlated with a good approximation of the predictions to the intended values. As stated before, this value is influenced by the network terms, so we utilize the output of the loss function to stir with the weights and bias values. However, how is this fulfilled?

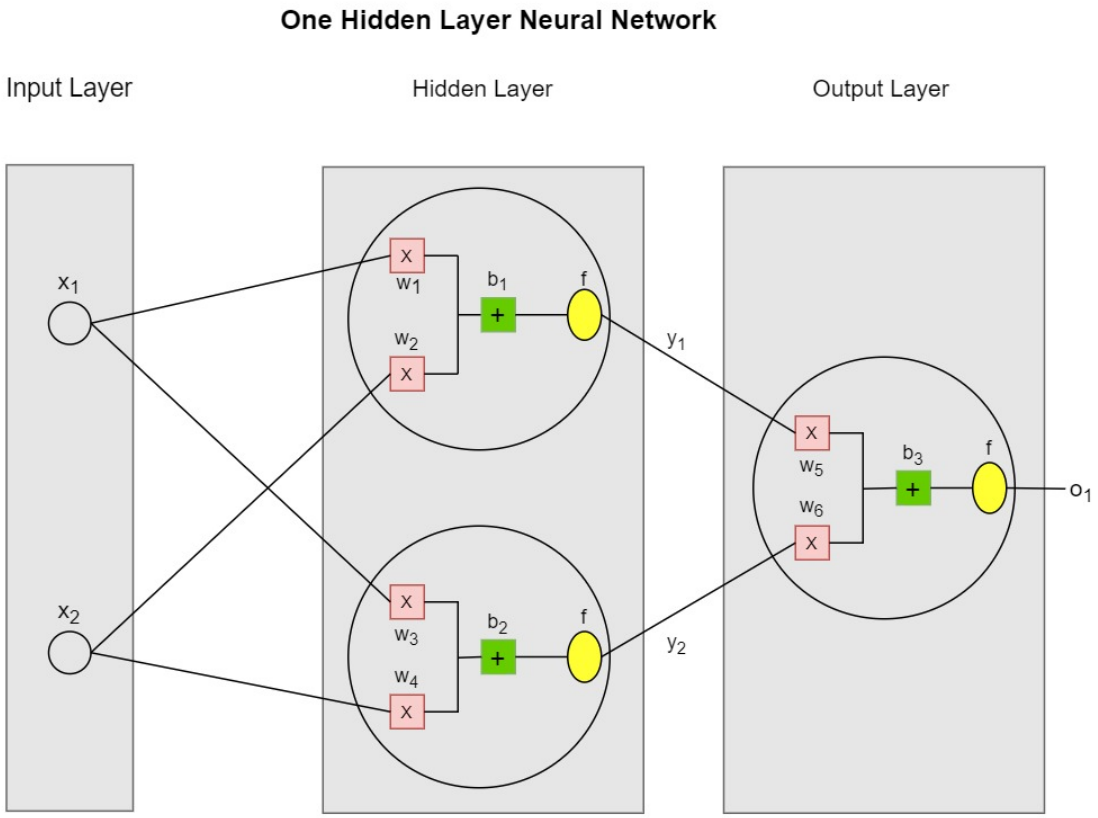


Figure 2.10: Scheme of a neural network presenting only one hidden layer which has two neurons, adapted from [45].

To begin, it is necessary to understand the two major instances that occur throughout the training process: feedforward and backpropagation. In a summarized way, the former is pushing the input data from the input layer until the output value is obtained, as previously demonstrated, and the latter is the propagation of the loss function result all the way to the initial layers of the neural network. This backpropagation is what drives the adjustment of the weights and biases, by providing the information of how these parameters should be modified by the use of partial derivatives.

Considering the simple neural network presented in Figure 2.10, it is possible to express y_1 and y_2 as follows

$$\begin{aligned} y_1 &= f(x_1 * w_1 + x_2 * w_2 + b_1), \\ y_2 &= f(x_1 * w_3 + x_2 * w_4 + b_2). \end{aligned} \tag{2.4}$$

Thus, the network output y_{pred} is given by

$$y_{pred} = o_1 = f(y_1 * w_5 + y_2 * w_6 + b_3). \tag{2.5}$$

Using 2.4 and 2.5 these formulas, it is possible to perform the feedforward, after which the result from the loss function may be calculated. In order to comprehend

how the partial derivatives are used, let us calculate the formulas that demonstrate how modifying the weight w_1 will change the result of the loss function, or in other words, let us calculate $\frac{\partial L}{\partial w_1}$.

Using the mathematical chain rule, we can write

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1} \tag{2.6}$$

Firstly, we can define the first term $\frac{\partial L}{\partial y_{pred}}$ as

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial \left(\frac{1}{n} \sum_{i=1}^{n} (y_{true} - y_{pred})^2 \right)}{\partial y_{pred}} = \frac{1}{n} * \sum_{i=1}^{n} -2(y_{true} - y_{pred}). \tag{2.7}$$

Having in consideration that w_1 only affects y_1 and never y_2, it is possible to use the chain rule one more time to rewrite $\frac{\partial y_{pred}}{\partial w_1}$ as follows,

$$\frac{\partial y_{pred}}{\partial w_1} = \frac{\partial y_{pred}}{\partial y_1} * \frac{\partial y_1}{\partial w_1}, \tag{2.8}$$

where we have

$$\begin{aligned} y_{pred} = f(y_1 * w_5 + y_2 * w_6 + b_3) \quad &\longrightarrow \quad \frac{\partial y_{pred}}{\partial y_1} = w_5 * f'(y_1 * w_5 + y_2 * w_6 + b_3), \\ y_1 = f(x_1 * w_1 + x_2 * w_2 + b_1) \quad &\longrightarrow \quad \frac{\partial y_1}{\partial w_1} = f'(x_1 * w_1 + x_2 * w_2 + b_1). \end{aligned} \tag{2.9}$$

The f' represents the derivative of the activation function, and in this example, the sigmoid introduced before will be used, so we obtain

$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) * (1 - f(x)). \tag{2.10}$$

As it is, we have already defined all necessary terms to calculate how modifying the weight w_1 is going to affect the loss function, which can be represented as

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial y_1} * \frac{\partial y_1}{\partial w_1}. \tag{2.11}$$

To close, it is imperative to know how one should change the weight value by having the partial derivative value. For this purpose, an algorithm known as Stochastic Gradient Descent (SGD) may be used. A new and better value for w_1 can be obtained by performing the following operation

$$w_{1_{new}} = w_{1_{old}} - \eta * \frac{\partial L}{\partial w_1}, \tag{2.12}$$

where η is a constant named learning rate that, as the name suggests, regulates the pace at which the model terms change thereby controlling how quickly they learn. To

obtain the optimal value for w_1 this process is redone multiple times until the value given by the loss function is stabilizing, which ideally is approximately zero.

The usage of artificial neural networks has numerous different architecture approaches. We will now analyse some of them in the following chapters.

2.2.2.2 Feedforward Neural Networks

The most basic architecture in deep learning is a Feedforward Neural Networks (FNN), which is also referred to as a Multilayer Perceptron (MPL). It is made up of different neurons, with the previously described behaviour, which are generally organized in many hidden layers. The connections are organized according to two principles: a neuron from layer n has as input all the preceding layer neurons' outputs, and its output must be propagated as input to all of the neurons of the following hidden layer. It is important to note that this type of layer, known as the fully-connected layer, restrains any possibility of feedback in FNN's. Yuxi Li in [36], defines a FNN as the application of a certain mathematical function to the inputs, however, this so-called function is actually sliced into multiple much basic functions, each are distributed throughout the different layers. FNN's represent a type of memoryless system since in this architecture, the previous states of the machine have no bearing on the current one due to the information being propagated to the next layer, a process known as feedforward.

A simple example of a feedforward neural network may be seen in the schematic presented in Figure 2.11. Note that every neuron is linked to all preceding and succeeding ones, as it should be in a fully-connected layer.

2.2.2.3 Convolutional Neural Networks

The Convolutional Neural Network (CNN) is a variation from the FNN that, in practical terms, has been widely used mainly in image processing since it is able to cope efficiently with multi-dimensional vectors like those seen in images. This is explained mainly due to the fact that a CNN, when compared with a typical FNN with the approximated same size layers, is easier to train since it is usually made up of fewer connections and has fewer parameters to tune, therefore requiring less computational resources [21]. In CNN's it is possible to find not only fully-connected layers but also pooling and convolutional layers, the former being a layer that separates an input into multiple blocks and then retrieves a certain value from it, such as its maximum or average, and the latter being a slight variation from the fully-connected ones, that uses mathematical convolutional operations, but not requiring that a neuron of a certain layer propagates its output to every neuron on the following hidden layer.

According to Pariwat Ongsulee [30], the major reason why CNNs obtain better results than FNNs is the fact that when one layer is considered, the value utilized for the weight of any neuron on that layer is the same. This inevitably correlates to a greater generalization

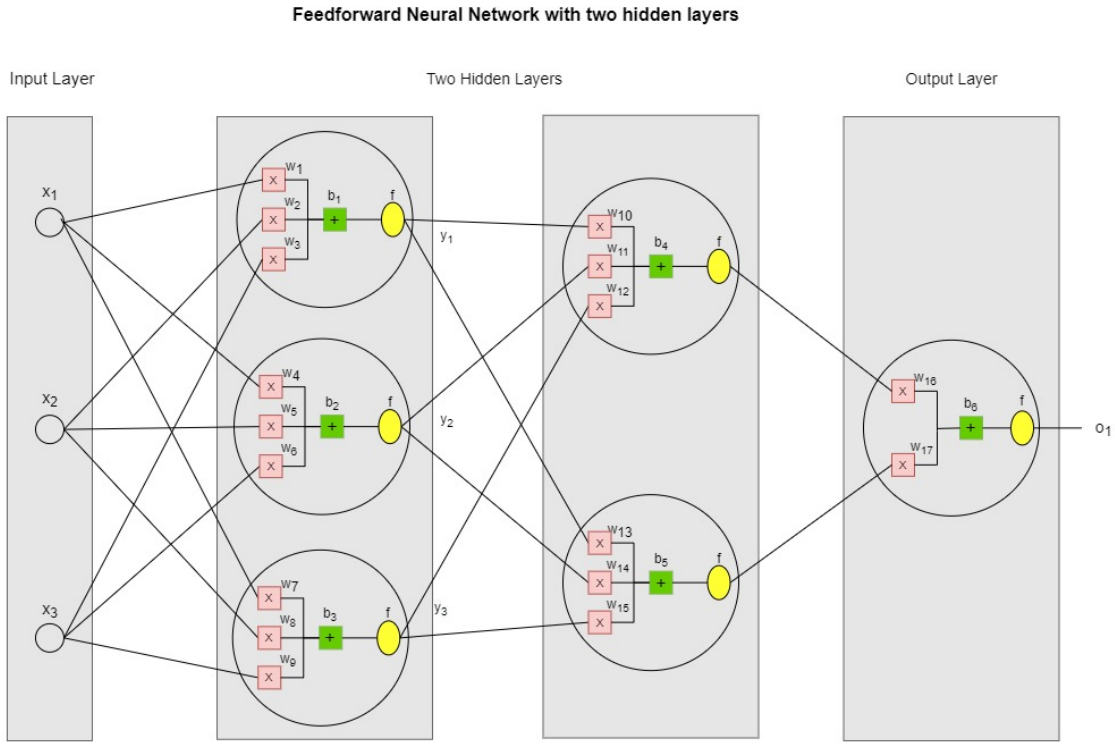


Figure 2.11: representation of a simple feedforward neural network which is composed by 6 neurons, with only two hidden layers, adapted from [4].

power avoiding overfitting, and reducing the number of parameters substantially to tune, thus confirming its superior estimation performance.

2.2.2.4 Recurrent Neural Networks

Another possible architecture in deep machine learning is the Recurrent Neural Network (RNN), which is a multilayer network primarily designed to address the sequential data scenarios, which are scenarios where the time sequence or order of the input signals has an impact on the model's training. In this type of architecture is already possible to find loops in the flow of information if a certain layer receives as input its own activation function result, meaning that feedforward is not the only propagation possible to find. This results in an architecture that supports memory, allowing the model to identify temporal correlations, unlike the previous architectures.

In an RNN, the input dataset is partitioned into time-step blocks rather than being supplied to the model as a vector as in prior models. Then, each of these blocks is the input delivered to a certain layer in the network. Typically, each layer has two inputs the first one being its corresponding time-step block and the second being the intermediate value calculated in the previous layer, or in other words, the preceding step's hidden state.

It is imperative to note that although fairly simple and valuable, RNN presents itself

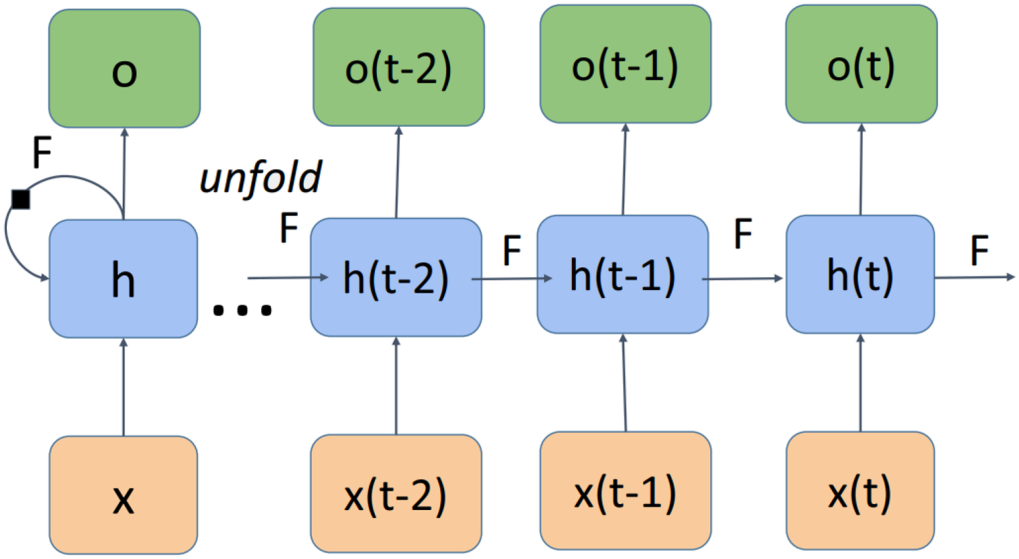


Figure 2.12: Scheme of simple recurrent layer where is possible to observe the partition of the initial input data, x, and the different inputs in every neuron. Adapted from [4].

with a few disadvantages as well, particularly the vanishing gradient problem, and by opposition the exploding gradient as well, which make it a hard model to train [31]. Let us recall that the models use the value calculated by the loss function to tweak the parameters' values and train the network. The vanishing gradient problem refers to the fact that, when backpropagating this value to the different layers of the network, this value slightly decreases in each layer, resulting in this value being very poor when it finally reaches the first layers. This means that these first layers' parameters will almost not be modified, therefore not learning, which then will create a long-term memory problem. However, the last layers will not suffer from this, which implies that short-term memory is not to be worried about. The exploding gradient problem is the exact opposite situation, where the backpropagation value keeps getting higher at each iteration, therefore being very close to one once it reaches the initial layers, which will cause over-sensitive training of these layers' parameters.

2.2.2.5 Long Short-Term Memory Networks

Long Short-Term Memory Network (LSTM) was the architecture designed to overcome the vanishing gradient problem that affects the long-term memory performance of RNNs. In order to improve the capabilities of the model's long-term memory, it is necessary to provide the ability to adequately select how much influence a piece of information has on the future states of the model while having in consideration to maintain the short-term memory performance in the process. As a result, LSTMs were created to integrate nonlinear controls of what information to forget or keep into the RNN architecture, based on the same information presented to it [38].

Let us address how LSTMs can achieve this. It is essential to understand that in this

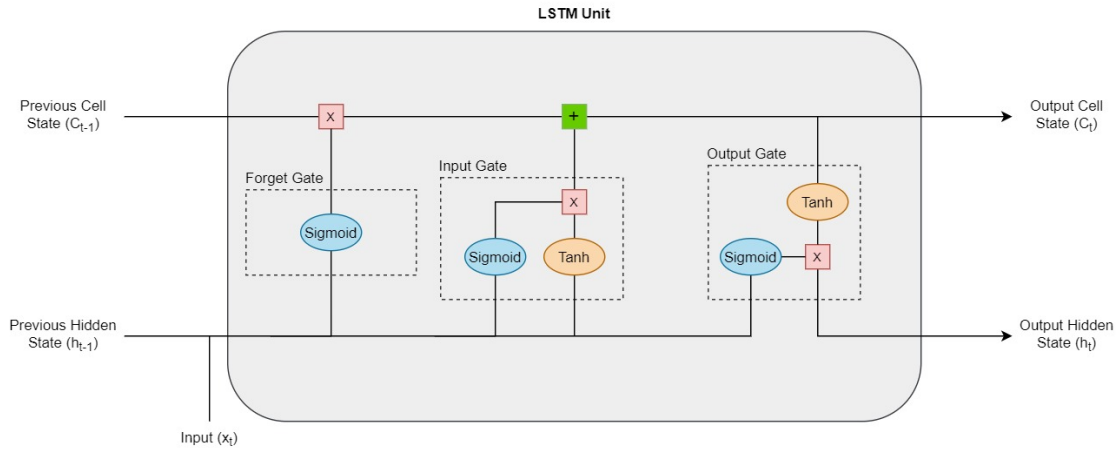


Figure 2.13: Representation of a typical repeating unit found in LSTM networks, adapted from [26].

architecture, the hidden state and the current cell state are propagated to the following layer, in addition to the specific layer's time-step input. Then there are three separate gating units in each LSTM unit, as represented in Figure 2.13, that determine how much of both of these inputs is stored or forgotten [26].

- Forget gate (f) – is responsible for controlling how much of the preceding cell state information is kept, and it does this by concatenating its time-step input, x_t, with the hidden state from the previous time-step, h_{t-1}, and then applying a *sigmoid* function to it. The extreme cases are a complete deletion or a complete copy of the previous state, being the *sigmoid* result for each 0 and 1 respectively

$$f = \sigma(concatenate(h_{t-1}, x_t)). \tag{2.13}$$

- Input gate (i) – this is the gate that manages how much information should affect the calculation of the current cell state update. It uses the same concatenation of the signals as the last one, but this time it separates it into two separate samples, where to one of them it applies the *sigmoid* function and to the other the tanh function, multiplying both at the end, equivalent to

$$i = \tanh(concatenate(h_{t-1}, x_t)) * \sigma(concatenate(h_{t-1}, x_t)). \tag{2.14}$$

  It is also important to note that from now, it is possible to calculate the current cell state, C_t, which will also be needed for the last gate, given by:

$$C_t = (f * C_{t-1}) + i. \tag{2.15}$$

- Output gate – determines the feed back given to the system's next layer, or in other words, calculates the hidden state, h_t, that is going to be passed on. It uses the same

Figure 2.14: Attention model functioning, with the relevant features highlighted, where an example of output might be a mug standing on a wooden board. Adapted from [39].

concatenation of the input with the previous hidden state and applies a *sigmoid* function to it. Then it applies a tanh to the current cell state, and multiply both values, being the current hidden state given by

$$h_t == \sigma(concatenate(h_{t-1}, x_t)) * \tanh(C_t). \tag{2.16}$$

2.2.2.6 Attention Networks

Attention Network (AN) are another type of recurrent neural networks that open up exciting new possibilities for deep learning. Due to their lack of reliance on dimensional characterizations, ANs have shown to be highly suited for scenarios involving sequential data such as speech translation and image caption generation. These networks are able to partition the data, relying only on some parts of data each time to produce the output, which is significant because, as data sets utilized in machine learning get more complex, the output structure is often non-linearly dependent on the input. Since discrimination of the input dimension is not necessary, we may say that an attention network has a variable-length memory, being appropriate for scenarios where it may be beneficial to have a varying input dimension [20].

The ability of human eyesight to focus on some characteristics while diminishing others is the motivation for this type of network concept. If someone saw a photograph of their morning coffee mug, they would probably identify it as precisely it, a simple mug. However, they would be undermining the table where the mug is, the teaspoon inside it, or other irrelevant aspects of the picture. This is the core principle of these networks, and it is accomplished by dynamically focusing exclusively on particular features of the input at any given moment, as represented in Figure 2.14.

The key issue with this approach is determining which sub-sequences of the input to focus on, and it may be solved in two ways:

- Hard attention mechanism - is a stochastic process, where the attention value denotes the probability of a certain sub-region of the input to be selected to produce the output. As the term "stochastic"implies, given a certain scenario, several possibilities can be chosen, resulting in a variety of possible outcomes, which may differ if run multiple times.

- Soft attention mechanism – is a deterministic procedure that computes the same result every time, in contrast to the former. Soft attention employs weight factors to determine which regions to focus on, allowing it to focus on numerous locations simultaneously.

The primary catalyst of the Attention Networks' development is that, despite bringing considerable improvements in the long-term memory capabilities over the RNNs, the LSTMs were still struggling with information that had been seen many time-steps before. Attention Networks are a more feasible solution for dealing with this problem, as proven by the authors in [34]. They demonstrated that it could cope with short-term and long-term memory, with astonishingly similar prediction accuracy rates in both circumstances.

2.3 Network Management using Mobility Analytics

The employment of machine learning also brings another major upside in networks that is the ability to monitor and analyze data generated from within the network, denominated as network-level data, or its end devices, known as app-level data, which before was unfeasible and now may bring enormous benefits to its management [43]. Usually, deep learning is the way to go for scenarios where there are lots of types of previous data and when the data is a product of different sources. Therefore, deep learning is usually a good option for this analysis since mobile data is plentiful and heterogeneous. However, as demonstrated next, most past works do not employ deep learning techniques. As said, this analysis brings new opportunities to the table in terms of network management, where some examples are referred to in the following sections. These examples are summarized at the end of the chapter in Table 2.1, and some few other examples can be found [14].

2.3.1 Routing

The information gathered during the mobility prediction phase can greatly improve the routing procedure on networks that include non-stationary nodes. This potential to improve routing can be seen by the analysis of parameters such as data replication, packet loss percentage, and delivery delay.

It is important to note that having more knowledge, or more control if that is the case, of the entities' movements, typically reflects in higher probabilities of the data reaching its final destination as stated in [5]. In direct data delivery, if the entity mobility is random, there is a high chance that the data will not reach destination B, and in the other extreme, if the entities' movements are controlled, its delivery is almost guaranteed. Being able to predict the entities' mobility may help to understand whether there is a high probability of delivery to the destination or not, given a determined entity. The same principle applies to indirect data delivery on both synchronous and asynchronous methods when choosing which entities and nodes one should transmit the data to. In the case of indirect delivery, there is another aspect to account for, which refers to the data replication, where an entity, while passing the data to another, may choose to keep a copy on its buffer or discard it. Higher amounts of replicated data among the network nodes will result in higher delivery rates at the expense of available bandwidth and storage. Here, the mobility prediction may improve the network management as well by making an informed decision to replicate it or not, instead of always or never replicating the data.

When the nodes in a network are mobile, it is also important to adopt new metrics to decide the best path. As explained in [19], in legacy networks, the shortest path is an excellent theory to follow, but not in networks with mobile nodes because being the shortest path is not equivalent to being the most stable one. Therefore, the authors of [19] propose using a mobility prediction algorithm based on a three-layer RNN to determine the most stable path, specifically in an ad-hoc network. Having prior knowledge of the entities' movements provided by the RNN allows the system to calculate the Path Expiration Time (PET) of the various possible paths, therefore representing their stability. This provides information about how much longer a communication link will be operational, resulting in lesser rates of lost data, enhancing the network' Quality of Service (QoS) by improving its routing capabilities.

2.3.2 Location-Based Applications

It is also feasible to use the data obtained from the mobility prediction model to create additional applications that will benefit the network infrastructure in some way through the determination of the users' locations.

The authors of the study done in [7] present an excellent example of a location-based service by presenting an alternative to Global Positioning System (GPS) through mobility prediction. GPS is a system that consumes a considerable amount of energy, which is problematic for mobile devices since most of them rely on limited batteries. With this in mind, this work demonstrates that by the usage of a second-order MC model in Android phones, it is possible to perform location prediction with higher accuracy than the GPS. The average energy savings were analyzed along two different paths, resulting in savings of 58.9% and 65%, respectively. Furthermore, by utilizing this type of service instead of the common GPS there will also be an alleviation of the network's bandwith by removing

GPSs control packets.

Another possible use of location-based services can be found in [22], where it is proposed a mechanism for anticipating crowd formations. In cases where massive crowds gather in a certain location, is not uncommon for the networks to drop users' connection requests or to be noticed a reduction in Quality of Experience (QoE). This work proposes a mechanism that, by performing mobility prediction, can anticipate this gathering and accommodate resources to cope with it. This can be done by initializing load balancing mechanisms or even activating a smaller cell in the predicted area of the crowd formation. Despite the gains that this method may bring, it is important to note that for mobility prediction, the authors opted only for a simple geometrical system that relies on aspects such as direction and velocity of movement that may not be suitable for more complex scenarios.

2.3.3 Load Balancing

Another important management function obtained is load balancing. It refers to the ability to adapt resource utilization in a dynamic manner while taking into account the current state of the network.

In [11], the authors focused on Self-Organizing Networks (SONs). They propose a MC model to predict the users' mobility. With this information, it is not only possible to allocate resources in the predicted next Base Station (BS), but is also possible to predict which BS are going to be more fraught on traffic and therefore perform informed management, with tasks such as exchanging the BS responsible for certain nodes.

On the other hand, work [28] proposes a Proactive Load Balancing (PLB) to maximize the QoE as possible when users are present in cells where the serving BS is lightly loaded. They utilize a mobility prediction model similar to the one used in the previously discussed work [11] to infer the users' mobility. Afterwards, it analyzes if the BS that serves the predicted cell demonstrates a low traffic load, and when that is the case, it utilizes the available resources to pre-cache the users' future content to minimize the delay.

2.3.4 Pre-Caching

Pre-caching is an important tool as well. Caching is the mechanism where data is retrieved from its source in advance and then stored for future use. When used in conjunction with mobility prediction is possible to pre-cache information that is likely to be requested by the user in its future expected location.

In [40] the authors studied the ability to cache multimedia content in advance, specifically in vehicular networks. This is an effective way of enhancing the QoS in scenarios where storage and link stability are scarce because parts of the content stay ready in locations through which the user will pass by, more specifically in one of the Road Side Unit (RSU) deployed in the environment. Firstly, the user's mobility is predicted by means of a HMM model determining which is the most probable RSU to be in contact with the

user, and afterwards, these chunks of content are cached in advance into this RSU. As demonstrated by the authors, this results in an incredible decrease in delivery latency times.

However, the previous work [40] assume that the inference made by their respective mobility prediction models is always right, but that is not realistic. Furthermore, in general, models with higher predictive accuracy represent more computational complexity to the system. Therefore, when pre-caching it is customary to introduce redundancy by pre-caching the chunks of content on more than one RSU, as it is the case of [40], which translates into non-optimal utilization of cache memory and bandwidth. Taking into consideration that Information-Centric Networks (ICN) have been appointed as the future of Internet architectures, the authors on [1] developed a system to work on ICN networks that aim at eliminating the need for redundancy. They propose predicting the mobility with a reasonably simple Markov based method, followed by understanding how reliable this prediction is. If not much reliable, authors advocate that the pre-caching should not be done in the edge device, the RSU, but instead on a device hierarchically higher in the network. This enables lesser overwhelming use of the networks' resources at the expense of lesser latency gains than the previous method.

2.3.5 Handover Management

Handover processes are known to deteriorate the QoE since the user may notice sudden drops in service quality. Mobility prediction is capable of tackling this problem as well, by transforming them into more efficient and non-disturbing procedures.

An example of this can be found in work [27], which intends to determine what is the most suitable BS to which a user should connect, having this decision supported by machine learning. For this work, a MC model was used to predict the users' mobility to determine suitable BS, but the ultimate decision is taken while taking into consideration the strength and quality of the received signal.

In [41], the authors developed an enhanced Handover Mechanism using Mobility Prediction (eHMP), that also helps at handling handover mechanisms between not only Wi-fi networks, but between Wi-fi networks and cellular ones as well. In the first moment, the authors utilize a HMM to perform user mobility prediction. After the prediction is made, the eHMP has sufficient information in order to optimize the handover process itself, resulting in considerable gains in terms of the percentage of packets that needed re-transmission and in the network's overall throughput. It is also relevant to note that eHMP, in terms of the network's throughput gains, did exceptionally well in scenarios where the handover was done only between Wi-fi networks in comparison to scenarios between different types of networks.

Another case where mobility prediction can improve the handover management is when the users are travelling close to the frontier between two different cells, served by

different BSs. This scenario often results in having the user's device engaged in intermittent handovers between the two BSs, known as the ping-pong effect, resulting in higher delays, higher energy consumption, and high resource waste. The work in [13] developed the Vehicular Location Prediction Handover Algorithm (VLPHA) that is able to resolve this problem. VLPHA takes into consideration the user's next location predicted via a MC model and the symbols' quality of the received signal from both the current cell and the predicted one. If both symbol's quality is similar, VLPHA considers the handover unnecessary, therefore protecting the network of the ping-pong effect.

2.3.6 Resource Allocation

Mobility prediction also introduces new possibilities in terms of resource allocation. In environments where resources, such as bandwidth and available memory, are getting scarcer, it is vital to control them intelligently.

By the analysis of this data is also possible to enhance network resource management by their pre-allocation. Works [33, 2] research on the capability of pre-allocating resources in mobile networks. P.Pratap and P. Agrawal on [33] developed a second-order Markov Chain, which is a variation from the typical MC described in Subsection 2.2.1.2, that differentiates itself from the former by making its prediction based not only on the current state but in the last one as well. With this algorithm, the authors were able to predict with reasonable accuracy which BS the user will most likely connect. This information enables the pre-allocation of resources, such as bandwidth, on the predicted BS, if available, therefore resulting in an almost unnoticed handover, minimizing its impact on the QoS and decreasing the probability of handover failure.

Worthy of a mentioned as well is the work [23] that analyzed in-depth the shadow cluster concept, which is optimal to ensure the delivery of certain QoS's levels in scenarios where the network' cells are small, therefore resulting in high volumes of handovers. This concept aims towards dynamic management of the network resources, especially bandwidth, and its goal is to optimize the resource utilization to its maximum while still reserving sufficient resources to maintain the rate of unsuccessful handovers below a defined threshold. Its effectiveness was proved in the paper by bringing the percentage of dropped calls from 14% all the way down to 1%. This concept is also dependent on prior knowledge of the user's mobility to predict the required resources at any given time. However, the results of this work must not be taken in an absolute way since the mobility prediction in this work is done in an oversimplified way by only considering highway scenarios where the users move at a constant speed in one of two possible directions. Despite being a promising idea, it requires further proof with more realistic data.

Table 2.1: Services made possible by mobile data analysis

Type of Service	Type of Mobility Prediction	Work's Objective	Reference
Routing Improvement	Recurrent Neural Network	Determining the optimal route through PET calculation	[19]
Location-based Applications	Second-order Markov Chain	Locates the user through mobility data instead of the energy consuming GPS	[7]
	Mathematical system	Predicts the formation of crowds in a certain area, being able to then initiate a load balancing protocol	[22]
Load Balance Management	Markov Chain	Is able to foresee traffic congestion in a certain base station	[11]
	Markov Chain	Analyzes the traffic load, if soft then utilizes pre-caching to optimize QoE with the unused resources	[28]
Pre-Caching Management	Hidden Markov Model	Utilizes the mobility prediction to perform pre-caching on the next RSU to which the user will connect to	[40]
	Markov Chain	Proposes to perform the pre-caching not only on edge devices, which allows to eliminate redundancy	[1]
Handover Management	Markov Chain	Determines the next BS a user should connect to, depending on the mobility prediction and signal received from candidate BSs	[27]
	Hidden Markov Model	Optimizes the handover process to achieve fewer levels of dropped users	[41]
	Markov Chain	Prevents against the ping-pong effect by utilizing the mobility prediction and the quality of the signals received from both BSs	[13]
Resource Allocation	Second-order Markov Chain	It performs the pre-allocation of resources such as bandwith, in handover situations	[33]
	Simple mathematical system	Proposes the shadow cluster concept, that aims at bringing the resource management of a BS to the more efficient state possible	[23]

2.4 Summary

During this chapter we performed an extensive survey on studies previously developed by other authors focused on the mobility prediction and its application. Firstly, we explained why the current networks will not be suitable in the near future. Then, we demonstrated how utilizing the mobility of non-stationary entities may be a viable approach to tackle the problems enunciated. Since mobility prediction is a crucial tool in this process, we then proceded to study various machine learning techniques. This study will support the choice of some approaches over others. Lastly, we presented multiple studies performed that showcased different services that are able to improve telecommunications network management through the acquisition of mobility information. Throughout the next chapter, we will explain how the mobility prediction is performed, and present the obtained results with two different approaches.

3

Mobility Prediction

Throughout Chapter 2 we have explained why it is so crucial to consider the mobility of the various devices that compose a modern network, as well as enumerating a few examples of how the knowledge of the mobility may be beneficial to the management of the network itself. We also surveyed different techniques that can be utilized to perform mobility prediction. However, we did not address how exactly these methods may perform such a task. This chapter presents two models used to predict mobility, where the first one is an evolution of the work in [16], and the second is a model proposed by us. Beyond this, we also compare the performance of both models, and study the influence of the sequence length and of the spatial-sampling resolution on the model's performance.

It is important to understand that predictions can be distinguished into two categories. The first one is known as short-term prediction, and it refers to the prediction of only one future location (or spatial sampling). The other is denominated as long-term prediction, and it is related to the prediction of multiple locations (or multiple spatial positions). Usually, the latter requires more prior information since the prediction is being made for a time instance further away. In our research, we opted to assess the model's performance in both cases, as well as how it evolves as the number of predicted locations rises. Therefore, we decided to develop the prediction models based on deep learning approaches since they are known to cope better with long-term predictions. The authors of [17] demonstrated this, where a deep learning algorithm outperformed a machine learning one not only in terms of accuracy but especially in the computational time required.

Firstly, in Section 3.1 we explain the approach used on the prediction problem, which can be applied to any similar scenario. In Section 3.2, we describe not only the development of the models utilized to perform the predictions, but also the method employed to validate these models. In Section 3.3, we perform an analysis of the specific dataset utilized by to train the models, which can be viewed as an indicator of how well the models are going to adapt to the data. And lastly, in Section 3.4, we validate the models and interpret the obtained results concerning the models' accuracy, allowing us to conclude their prediction capacities.

3.1 System Model

Our goal is to develop a model that can receive a certain number of previously known locations and output a prediction concerning future ones. The methodology we developed is rather generic, meaning that it can be adapted in order to function in different environments in various cities. The concepts introduced next are also valid for alternate scenarios.

To construct our model we assume that a delimited region throughout which the entities´ movements (e.g., taxis trips) are monitored. Viewing this region as a whole, allows us to consider its divison into cells. These cells can take any size, as long as they maintain a constant size. The cells are later taken as the possible states for the entities' locations in the inputs and outputs of the model. The total number of existing cells, N, is arbitrary. However, one should know that while having many cells will result in describing the mobility with more detail, it will also hamper the predictions since more possible states are introduced.

A trajectory represents the path taken by any of the entities during a certain time frame. Therefore, after determining the N value to use, a trajectory is a set of consecutive cells visited by the mobile entity, where each cell is numbered between 1 and N. Typically, a dataset is composed of multiple records of the entities' known locations that together form various trajectories.

When building a prediction model it is necessary to define a fixed length for the various data inputs. For this reason, it is necessary to further partition each trajectory into a set of sequences. The sequence length, Λ, can take different values, however, once decided upon, all sequences have the same length. For any predetermined Λ value any sequence can be referenced as $S_m =< c^1, c^2, ..., c^\Lambda >$, where $c^1, c^2, ..., c^\Lambda$ correspond to the sequential cells that compose the sequence S_m.

Figure 3.1 illustrates an example of a trajectory in a scenario where N=16. The trajectory traverses 6 cells. Assuming $\Lambda = 4$, we obtain 3 sequences of 4 cells, specifically $S_1 =< c_{16}, c_{14}, c_9, c_6 >$, $S_2 =< c_{14}, c_9, c_6, c_7 >$ and lastly, $S_3 =< c_9, c_6, c_7, c_8 >$.

Let us assume a sequence composed by Λ cells. This sequence, as all others, can be divided into two parts. The first one has α cells, where α is the number of observable states or, in other words, the number of observed prior locations to take into account when performing the predictions. The second part has $\Lambda - \alpha$ cells, and it represents the number of cells whose location is predicted by the mobility model. Therefore the first part is the input provided and used by the model to predict the second. Note that these numbers should be adapted to the problem in hands, as they can greatly impact the model's performance.

One other concept that should not be overlooked, is the time sampling resolution. This equates to the periodicity at which the locations are gathered. If the time between samples is too long, it may result in vague information. If the sampling time is to low a high number of samples is obtained for very close locations.

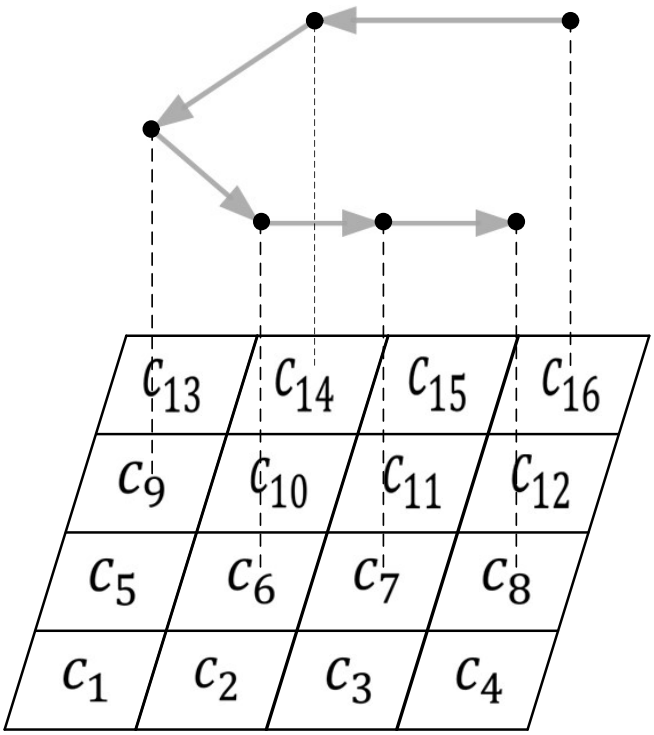


Figure 3.1: Example of a trajectory on an unspecified region, using a spatial resolution of N=16.

3.2 Prediction Methodology

In a first stage we developed an LSTM based model to tackle the scenario in hands since this approach is known to cope well with predictions for scenarios of such complexity, especially when long term predictions are performed since it is aware of the vanishing gradient problem. Later on, we also propose an ANN algorithm using the same data aiming to investigate if this model would also be capable of achieving good prediction performance, while demonstrating the expected need for fewer computational resources. The methodologies used for both methods are described in this Section, while the comparison between the achieved results is presented in Section 3.4.

The approach based on LSTM recurrent neural networks was the first to be developed, and so it is the first to be described. The model is composed by two sequential layers, the first one being an LSTM layer and the latter a dense one. The LSTM topology is illustrated in Figure 3.2. The LSTM layer comprises 16 hidden units, where each one of them is using the *tanh* as its activation function. The obtained values are then passed to the dense layer where the *sigmoid* is used as its activation function. For the model's training phase, we perform the training along 100 epochs with a batch size of 512 samples using the Adam optimizer and the categorical cross-entropy as the loss function. The configuration adopted is presented in Table 3.1.

In our work, we utilized a binary codification known as one-hot encoding [8], which

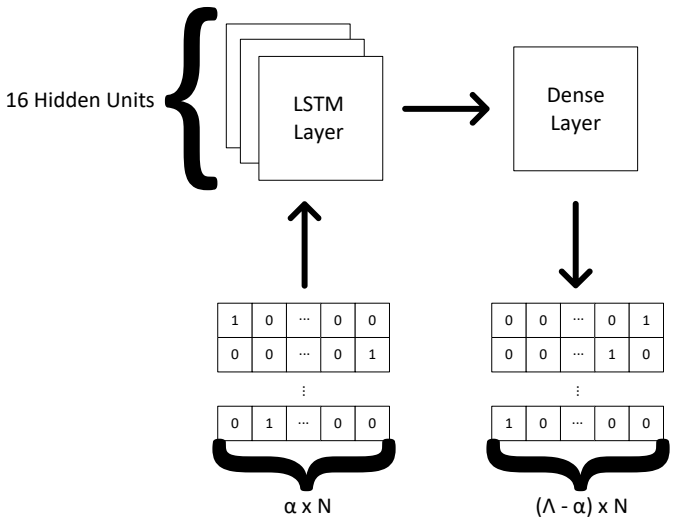


Figure 3.2: Structure of the LSTM model developed to perform predictions (adapted from [16]).

Table 3.1: Configuration of the LSTM model developed to perform mobility predictions (adapted from [17]).

LSTM Layer	Activation Function		Employed Loss Function	Utilized Optimizer	Number of Epochs	Batch Size
	LSTM Layer	Dense Layer				
16 hidden units	Tanh	Sigmoid	Categorical cross entropy	Adam	100	512

maps each variable to a binary vector where the different columns represent the possible features, containing 0's when the feature does not apply to the entry, and 1's when it does. One-hot encoding has the particularity of representing the possible states of a feature in an orthogonal way. In our scenario an entry relates to a determined cell and the columns to the possible cell numbers, therefore having N columns. In light of that, the model's input is a matrix with shape $\alpha \times N$, and the output matrix shape is consequently $(\Lambda - \alpha) \times N$.

When it comes to the ANN approach, it represents a simpler solution that we thought to be relevant since it could bring benefits in terms of required computational resources, altough it was also expected to not perform as well as the LSTM approach, specially if a similar topology was utilized. Consequently, we were bound to construct a model composed of multiple sequential layers, where we found that using 5 dense layers was a satisfying compromise in terms of complexity and required computational resources, as shown in Figure 3.3. All layers use Rectified Linear Unit (ReLU) as their activation function, except for the final one that utilizes the *softmax* limiting the outputs between [0,1]. In the training phase, the same 100 epochs and, 512 samples of batch size and Adam optimizer were used as in the previous method, however, the loss function employed in the training phase was the MSE. Once again, this configuration is summarized in Table 3.2. Note that, despite being composed of more layers than the previous approach, the ANN

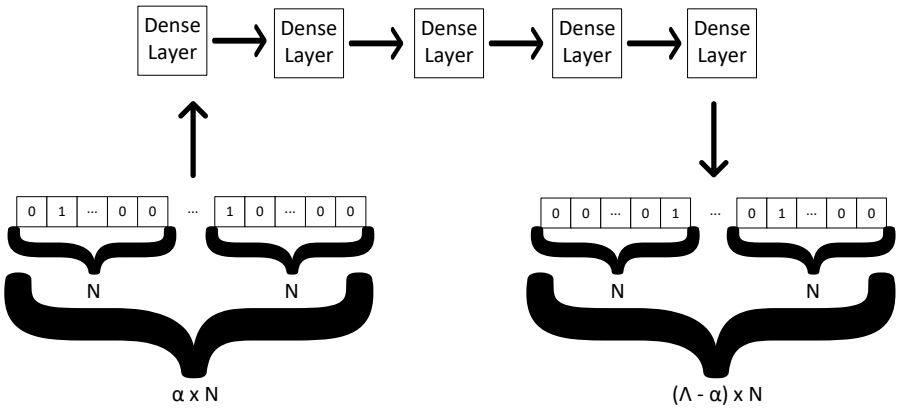


Figure 3.3: ANN based model's structure (adapted from [16]).

Table 3.2: ANN model's configuration (adapted from [17]).

Activation Function		Employed Loss Function	Utilized Optimizer	Number of Epochs	Batch Size
Initial Layers	Final Layer				
ReLu	Softmax	Mean squared error	Adam	100	512

model does not necessarily require more computational resources due to the simplicity of each dense layer when compared to a LSTM one.

In terms of used codifications, the one-hot encoding was also used in this approach. However, the ANN model expects a one-dimensional sequential input, which disallowed us to use a matrix, as previous explained, and forces us to translate the binary codification to a vector. Therefore, the input data for this model takes the shape of a vector with $\alpha \times N$ columns, and consequently, the output is also a vector with $(\Lambda - \alpha) \times N$ columns. The codification process is handled in the same way as before.

As explained above, the various trajectories acquired from the dataset require further partition into sequences. This will result in different partition results when using different sequence lengths. In order to analyze this aspect though, it is important to first distinguish two key concepts. When the partition of the trajectories is completed, we will obtain a set composed by the total of sequences that are product of this partition, which are used in the model's training process. Another obtained set is composed only by unique sequences, which means that no sequence can appear more than once in this set. The reason for us to consider the two of them is that in the first one it exists replication of sequences which is an important feature since it enables the model to be trained while taking into consideration the most probable sequences. On the other hand, when evaluating our model, using a set with sequence replication is inefficient because the prediction is deterministic, meaning that if one sequence is utilized for estimation multiple times, the result will always be the same.

Succeeding the training phase, after which the model is prepared to perform the intended predictions, we enter the validation phase, where the generated model is used to predict the $\Lambda - \alpha$ cells for each sequence of the unique sequences set. Completing

these predictions allow us to evaluate the model's Prediction Performance (PP), which is assessed by comparing each projected cell to the expected one as

$$PP = 100 \sum_{i=\alpha+1}^{\Lambda} p_{S_m} F(c^i_{pred}, c^i_{corr}), \tag{3.1}$$

where F is the function responsible for the direct comparison between the cell predicted by the model, c^i_{pred}, and the corresponding correct cell, c^i_{corr}, which returns 1 if the cells are equal, and 0 if they are not. Since we perform this validation with the unique sequences set where there is only one instance of each sequence, it is crucial to take into account which sequences appear more or less frequently in order to perform a balanced evaluation of the prediction performance. Therefore, the value returned by the comparison is then multiplied with the probability of the given sequence to occur in the data, p_{S_m}.

3.3 Dataset Analytics

The dataset utilized when constructing our models, which can be found in [10], concerns real-world information retrieved over the course of one full year, from GPS data acquisition systems implemented in 442 taxis that operated in Porto, Portugal. The use of this dataset is advantageous since every trajectory considered corresponds to a trip taken by a taxi driver, in opposition to synthetic datasets which are generated in an effort to emulate real-world scenarios. Furthermore, the considered data is particularly interesting due to the randomness of the trajectories adopted by taxis, which are much less predictable than personal cars or buses, adding another layer of difficulty to the mobility prediction scenario.

The considered region of the city represents an area with a total of 11.579 km^2 (2953 m height and 3921 m wide), which is to be divided into a grid composed by the determined number of cells of the same size, N, where the spatial-sampling resolution is introduced. Higher resolutions imply more cells while lower resolutions consider less cells in the same area of interest. Throughout our tests we considered three different numbers of cells, respectively, N = {16, 64, 256}. Since working with the GPS coordinates is unpractical, each coordinates entry is translated to the corresponding cell number before any work is carried out.

As previously stated, every trajectory need to be divided into sequences, that consist on a fixed length, Λ, while using a sliding window of one. During our work we have considered multiple sequence lengths, specifically Λ = {4, 8, 16, 20}, to further analyze the impact of the different sampling resolutions in different input lengths.

Furthermore, as stated before, we performed estimations for different numbers of predicted cells, which vary accordingly to each Λ value adopted, allowing us to assess the model's performance on both short and long-term predictions. While considering higher sequence lengths we are able to predict more cells accurately, and therefore, for

each $\Lambda = \{4, 8, 16, 20\}$ we propose to predict 1, 3, 4 and 5 cells, respectively. However, this is merely indicative and should be adapted according to the scenario's complexity and model capabilities. With this information we are able to classify each sequence as $S_m =< c^1, c^2, ..., c^\alpha, c^{\alpha+1}, ...c^\Lambda >$, where α represents the number of previous known cells considered. The remaining $\Lambda - \alpha$ cells represent the number of predicted cells.

Concerning the time sampling, the equipment records the vehicle current location every 15 seconds, and, as stated above, we consider this constant rate for both approaches. This means that, in the shortest studied time frame, which occurs when $\Lambda = 4$, each full sequence corresponds to the path taken by a taxi driver over the course of 1 minute. On the other hand, if the longest sequence length considered is employed, $\Lambda = 20$, each sequence will correspond to a path of 5 minutes.

In view of all the details enunciated before, the reader is now in full capacity to understand the two models implemented to perform the mobility prediction, as well as their validation. Analysing the number of total and unique sequences resultant from the trajectories' partition is a fairly good way of performing an *apriori* evaluation of the models' capabilities. Since the partition is done before the models are ever created, its findings apply to both the LSTM and the ANN models.

It is important to note that the different spatial resolution settings indicated before, $N = \{16, 64, 256\}$, are considered for both models, to enable a deep comparison between them. The same principle applies to the considered sequence lengths, $\Lambda = \{4, 8, 16, 20\}$, and to the correspondent number of cells to predict in each case, respectively 1, 3, 4, and 5 cells.

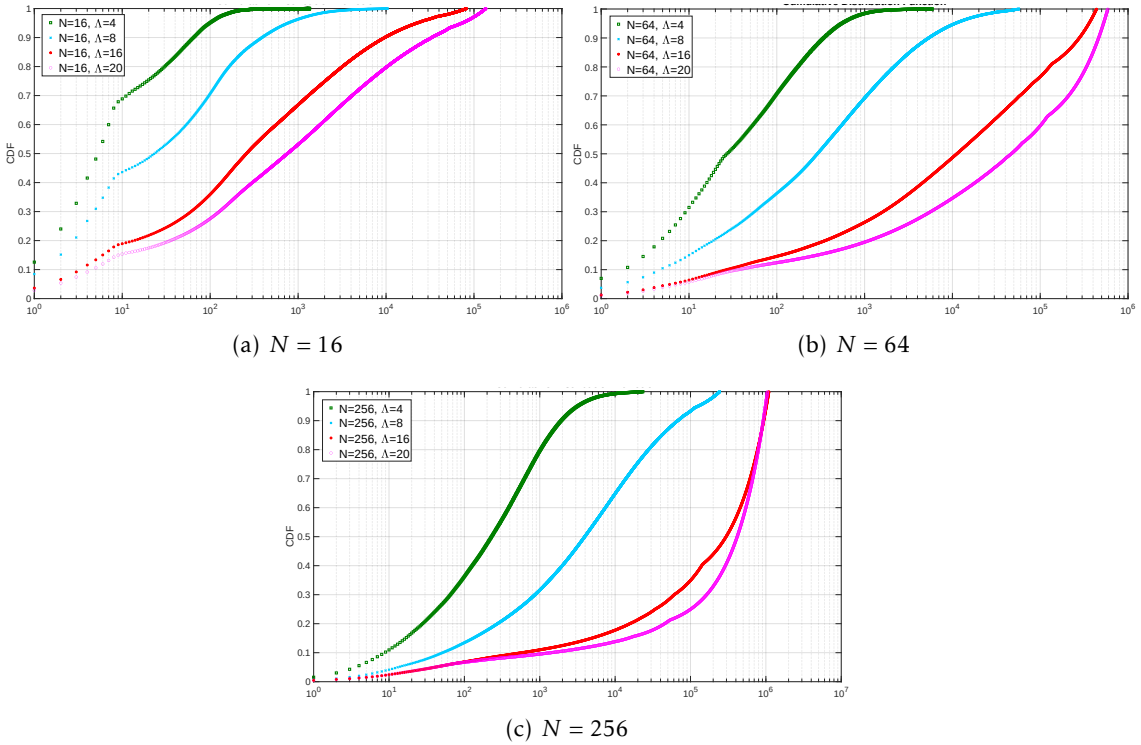
Since different combinations of Λ and N are used, we decided that it would be interesting to observe the behaviour of both sets for different combinations Λ and N. The results are indicated in Table 3.3. It is possible to observe that the increase of Λ will produce a lower number of total sequences which is expected since the sequence length increases. However, the number of resultant unique sequences actually increases since the resultant sequences are longer and consequently more descriptive of the entity's movement. When analyzing the sets due to different N parameters, it is also possible to conclude that increasing the number of cells will not affect the total number of sequences because the partition does not depend on the cell number while, on the other hand, the size of the unique sequences set rises drastically with its increase since the sequences became more capable of describing the entity's movement in a detailed manner, similarly to when Λ is incremented.

Another metric indicative of the model's performance is the Cumulative Distribution Function (CDF) of the sequences resultant from the dataset. As explained before, in the set containing the total number of sequences it is possible to find multiple instances of the same sequences, which means that there is more than one entry on the dataset corresponding to a taxi following that path. Our model settles on the likelihood of a sequence occurring, or in other words, the probability of the sequence, where sequences replicated more times represent more probable ones. When in the prediction phase, the

Table 3.3: Distribution of both sequence sets according to Λ and N.

N	Λ	Total Number of Sequences	Unique Sequences
16	4	2 752 580	1 349
	8	2 354 791	10 473
	16	1 598 190	82 082
	20	1 263 477	136 450
64	4	2 752 580	5 971
	8	2 354 791	57 225
	16	1 598 190	438 560
	20	1 263 477	588 711
256	4	2 752 580	23 340
	8	2 354 791	244 049
	16	1 598 190	1 095 666
	20	1 263 477	1 048 137

model essentially takes into account the prior α cells and looks for the sequence that is more likely to describe the entity's path. The graphic representation of the CDF of the unique sequences, which can be found in Figure 3.4, is a straightforward manner of analyzing these probabilities.



(a) $N = 16$ (b) $N = 64$

(c) $N = 256$

Figure 3.4: CDF of the unique sequences for different N and Λ values.

By analysing Figure 3.4(a) we identify that this is not a desirable scenario. For example, observing the case where $N = 16$ and $\Lambda = 4$, approximately 69% of the total sequence occurrences retrieved from the dataset relate to only 10 sequences. This means that these

10 sequences are much more dominant than the remaining ones, which demonstrates that the spatial resolution used is not adequate. By comparison, when analyzing the curve associated with the same Λ when $N = 256$, results plotted in Figure 3.4(c), the 10 more probable sequences refer to only 11% of the overall cumulative distribution. The scenarios with more dominant sequences are typically undesirable since they will result in less detailed and more repetitive predictions. However, they will also produce more accurate predictions. The spatial resolution has to be chosen while taking this into account, in an effort two balance the prediction accuracy with the relevance of the entities' mobility description.

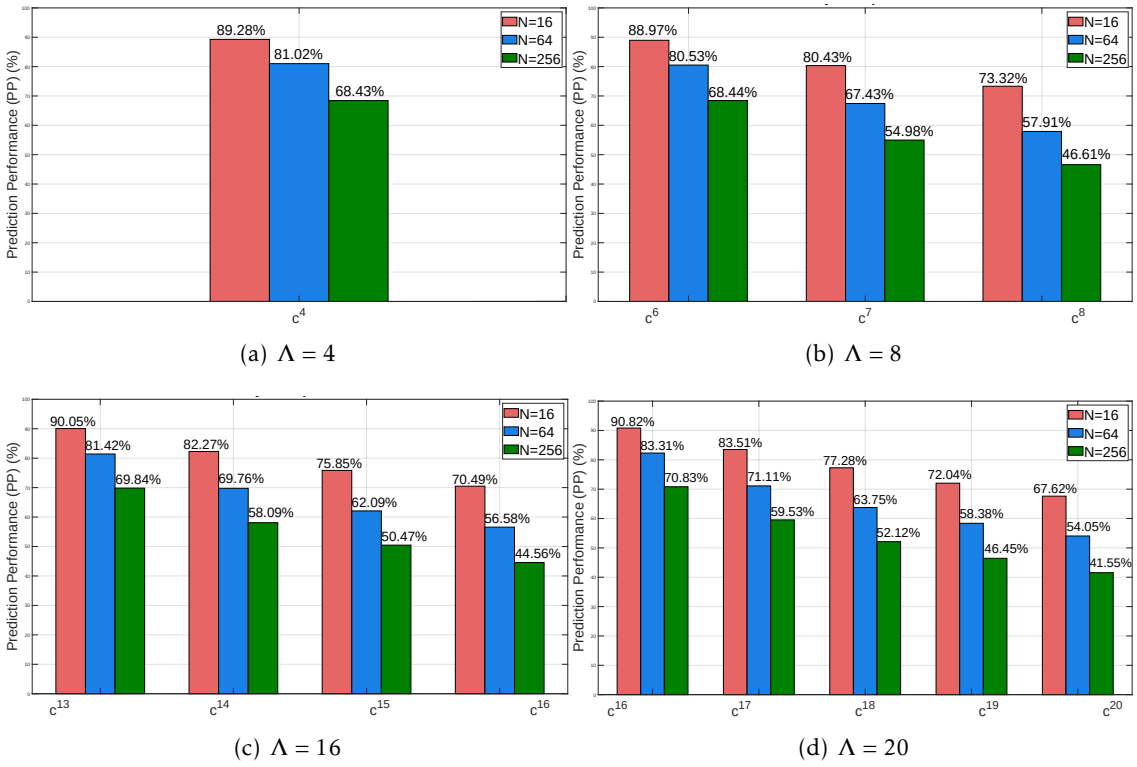
3.4 Performance Results

In this section, we present the results obtained for both the LSTM and ANN models not only concerning the prediction performance calculated as described in Section 3.1, but also in terms of required computational resources. It will be possible to analyze the behaviour of these metrics when the complexity of the scenario is increased, either by increasing the spatial resolutions, or by using a longer sequence length and consequently predicting a higher number of cells further away in the time perspective. Possible combinations of N and Λ were computed for both approaches, establishing a solid base for the comparison between them. It is important to mention that all the predictions were performed in an Intel 8-core i7-9800X @ 3.8 GHz with 128 GB of memory.

In a first moment we analyze the values obtained for the prediction performance with the LSTM based model, which will be followed by the comparison with the ANN one. The prediction performance values obtained for the LSTM approach are presented in Figure 3.5, separated accordingly to the different Λ values. Figure 3.5(a) considers $\Lambda = 4$ and refers to the short-term prediction of 1 cell. Long-term prediction scenarios are reported in Figures 3.5(b), 3.5(c), and 3.5(d), for $\Lambda = \{8, 16, 20\}$, which refer to 3, 4, and 5 predicted cells, respectively.

By analyzing only the short-term prediction results first, it is clear that the increase of the spatial resolution has a direct impact in the model's prediction accuracy, which is explained by the increase in the scenario's complexity. With more cells composing the grid the data becomes more diverse, creating less dominant sequences overall. However, it is important to highlight that despite the cost in terms of accuracy, since the cells are smaller, the model gains the capability of better describing the entities' movements with predictions more exact in terms of location, which may be beneficial for some applications. This feature remains present for long-term predictions as well.

When observing scenarios where more than one cell was predicted, the prediction performance difference between short and long-term predictions becomes evident. When considering any N and Λ values, it is possible to recognize that the more precise prediction is for the first cell, and its precision will decrease for the subsequent ones. Performing



(a) $\Lambda = 4$ (b) $\Lambda = 8$

(c) $\Lambda = 16$ (d) $\Lambda = 20$

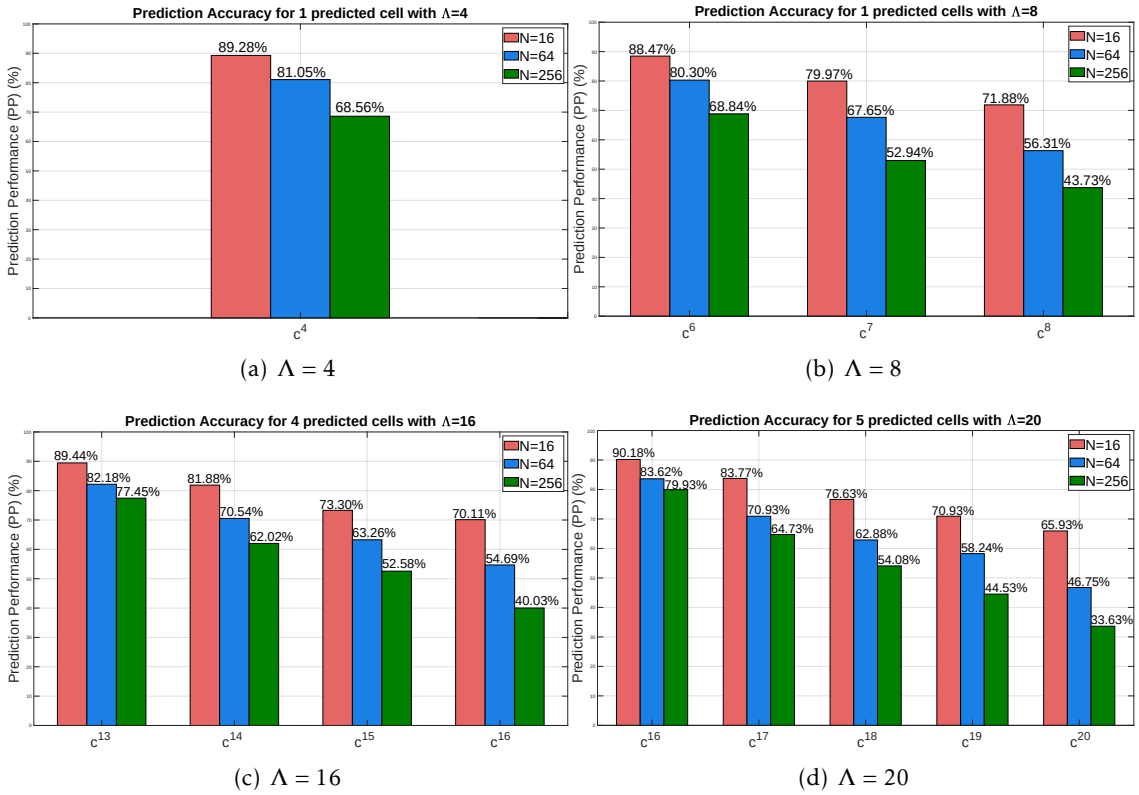
Figure 3.5: Prediction Performance results obtained for different N and Λ values, concerning the LSTM model.

predictions for time instances further away in a time perspective has demonstrated to be more challenging, which is expected since more uncertainty is introduced.

On another note, it is possible to confirm that the predictions become more accurate when more previous data is provided to perform them, as expected. When observing, for example, the first predicted cell across the different sequence lengths, it was shown that its prediction performance increases when the Λ value also increases, independently of the considered N. With the increase of Λ there also exists an increase of the number of previous cells, α, which represent more information to perform the same prediction. This principle also explains why for lower sequence lengths such as $\Lambda = 4$ it is irrelevant to consider multiple predicted cells, since the model does not own sufficient information about the entities past locations to predict future ones so far away in time.

On the other side, the prediction results obtained for the ANN based model are presented in Figure 3.6. Despite of what might be expected from a fairly simpler approach, the ANN model showed to be sufficient to describe the scenario in hands, demonstrating a prediction performance very similar the previous model. The accuracy results obtained are not only similar but, when high spatial sampling is utilized (N=256) and $\Lambda = 20$, even demonstrate an improvement of more approximately 9% for the first predicted cell . In contrast, the long-term prediction also decayed almost 8%, for the fifth predicted cell.

To sum up, in terms of obtained prediction performance results, both models are



(a) $\Lambda = 4$ (b) $\Lambda = 8$

(c) $\Lambda = 16$ (d) $\Lambda = 20$

Figure 3.6: Prediction performance with the ANN approach, for the various N and Λ values.

identical and very capable of performing the intended prediction in scenarios that reflect the same level of complexity required here. However, this similarity is not enough to take objective conclusions. An evaluation of the required computational resources during the prediction phase is also imperative to describe the correlation between the two approaches. In Table 3.4, we present the average time and its uncertainty range for each iteration, enabling a direct comparison of the required resources for both models.

Table 3.4: Computation times of both LSTM and ANN based approaches.

N	Λ	Computation time by iteration for LSTM (seconds)	Computation time by iteration for ANN (seconds)
16	4	0.0159 ± 0.0049	0.0147 ± 0.0075
	8	0.0165 ± 0.0050	0.0150 ± 0.0044
	16	0.0196 ± 0.0053	0.0150 ± 0.0028
	20	0.0208 ± 0.0054	0.0149 ± 0.0026
64	4	0.0156 ± 0.0049	0.0149 ± 0.0059
	8	0.0167 ± 0.0052	0.0148 ± 0.0042
	16	0.0199 ± 0.0062	0.0151 ± 0.0033
	20	0.0207 ± 0.0067	0.0152 ± 0.0032
256	4	0.0158 ± 0.0050	0.0150 ± 0.0054
	8	0.0167 ± 0.0055	0.0151 ± 0.0047
	16	0.0199 ± 0.0079	0.0155 ± 0.0044
	20	0.0208 ± 0.0080	0.0155 ± 0.0039

The results demonstrate a clear improvement of the ANN approach over the LSTM one

in terms of computation times, which was already expected due to the lower complexity of the ANN based model. The discrepancy aggravates more as the scenario gets more complex, especially when the the sequence length is increased since it causes a huge impact in the LSTM computing time. In contrast, the ANN approach computation time is more robust to the changes in the sequence's length.

Finally, we conclude that the ANN based approach seems more suitable to perform predictions in scenarios as complex as ours. This approach has demonstrated to be evidently faster, or less computationally demanding, while also exhibiting that it is capable of identifying the data's patterns as the previous model. On the other hand, the results obtained through the LSTM model gave us indications that it was more suitable for long-term predictions, showing that should become even clearer if more complexity was required.

On a final note, the ANN model proposed in this section was comprised of 5 sequential dense layers. One might think that the assumption of more than 5 layers would be sufficient to outperform the previous approach in terms of prediction performance, especially since it is less demanding even with 5 layers. However, this is not necessarily true. We have confirmed that introducing more layers does not necessarily improve the prediction accuracy while increasing the computation time.

3.5 Summary

To sum up, in this chapter, we started by explaining on what a mobility prediction problem consists, and showcasing how we decided to handle it. This was followed by the demonstration of how we developed and validated the models that perform the mobility prediction. This approach was presented in a fairly generic manner, and should be taken into account throughout the subsequent chapters, since it is also applicable for the scenarios described in them. Then, we started to present the work performed more specifically to the considered scenario, in Porto's city, by analysing in depth the dataset. This analysis elucidates on how capable a model will be to learn the features necessary, by being trained with the dataset in question. To finalize, we presented the configuration of the developed mobility predictors, and compared their prediction accuracy, as well as the required computational resources. In the next chapter, we will demonstrate how information concerning mobility may help to improve communications network management.

4

Network Enhancements

Until now our work has aimed at describing, implementing and studying mobility prediction problems. The previous chapter presented two models that can be used for mobility prediction, as well as their prediction accuracy when applied to a vehicular dataset. However, and having into consideration the objective of this dissertation, it is still lacking some sort of demonstration of how mobility prediction may help to bring enhancements to managing a communications' network. That is the main focus of the current chapter.

With that said, we showcase the improvements obtained on a communications network management level by demonstrating the enhancements that mobility prediction can bring in two different scenarios with distinct mobility profiles: the first one running in Porto and presented in the previous chapter, and the second is based on traces collected from public buses traveling in the city of Aveiro, Portugal. In a scenario where a pre-caching mechanism is implemented, when the user requests a chunk of data, there are two possible outcomes: the mobility prediction is performed correctly, and the respective data is stored in the Road Side Unit (RSU) to which the end-user is connected, or the prediction is not accurate, thus requiring it to be downloaded from the Cloud once again. We will simulate the data's download time from the user's perspective by getting the Round Trip Time (RTT) of multiple packets sent to the nearby RSU and to the Cloud, which allows us to study the data retrieval time in both possible outcomes. We analysed the obtained RTTs and used them to show the enhancements that a pre-caching system could bring to a telecommunications network management, specifically in the considered scenarios.

In Section 4.1, we introduce the two mobility scenarios studied in this chapter, and highlight their differences. Afterwards, in Section 4.2, we perform the pre-prediction analysis in order to characterize the datasets. In Section 4.3, the prediction methodology is presented, describing both the prediction models as well as the Prediction Performance (PP) evaluation method. Finally, in Section 4.4, we show how the output of a mobility prediction model can enhance the performance of a communication's network, for example, by the means of a pre-caching mechanism.

4.1 Considered Scenarios

As stated above, in this chapter we consider two different scenarios. The aim of assuming two scenarios is to better demonstrate the enhancements brought by mobility prediction to a communication's network. On one hand, the obtained results will become much more relevant by proving their validity in more than one scenario. On a different note, the data acquired from this second scenario is very different from the previous one. In the city of Porto we are confronted with a challenging prediction scenario due to more unpredictable movement of the vehicles considered in the dataset, caused by taxis trips. Nevertheless, as seen in the previous chapter, we have a large amount of data. On the other side, this new scenario presents itself as a less bold prediction problem since the dataset results from data retrieved from buses, which obviously describe more deterministic trajectories, and therefore easier to predict. However, this dataset has low amounts of data which also shows the models' capability to adjust and make correct predictions.

The first dataset that we take into account is the one already presented in Chapter 3, so it is recommended to the reader to refer specifically to Section 3.3 for a better understanding of the scenario.

Despite utilizing the same dataset, an important distinction has to be made. In the previous chapter we handled the entire data as one block, using it entirely to fulfil both the model's training, as well as to perform the subsequent predictions to validate the model. However, this manner of validation lacks some realism, especially when considering the fact that in real-world scenarios, new trajectories never seen before may be necessary to be predicted, as happens with this approach. Thus, in this chapter, we are going to approach the mobility prediction validation differently from the previous chapter.

Throughout this chapter, we are going to divide the total available data into both the training dataset used exclusively to train the model, and the test dataset which is then used solely to validate it. This approach adds an extra layer of realism to the study performed on these scenarios.

As enunciated in the previous chapter, the information concerning Porto's taxis data is the result of 1 year of recording their GPS coordinates. Here, we will perform the aforementioned separation in such a way that the training dataset corresponds to 75% of the total trajectories, approximately 9 months, and where the test dataset relates to the remaining 25%, which translates into approximately 3 months of retrieved data.

It is also crucial to note that in Chapter 3 we analysed different combinations of both spatial sampling resolutions, $N = \{16, 64, 256\}$, and sequence lengths, $\Lambda = \{4, 8, 16, 20\}$. Since we have already studied how these factors affect the outcome prediction accuracy, we are not going to perform the same study here. Instead, we are going to use a spatial resolution of $N = 64$, and a sequence length of $\Lambda = 4$, therefore performing the prediction of only one future cell. From a time frame perspective, the adopted sequence length corresponds to 45 seconds.

The second dataset was constructed from the data retrieved from 10 buses that were being driven around the city of Aveiro, where there is an existing sensing, communication and computation infrastructure in place composed of 21 RSUs, dispersed over 5 km^2. Such infrastructure, entitled Aveiro Tech City Living Lab (ATCLL) [1], is open for experimentation by the research community, start-ups, entrepreneurs, etc., making available a vehicular network supported by the ITS-G5 and C-V2X, among other communication technologies, as described in [35]. The ATCLL ITS-G5 vehicular network is formed by On-Board Units (OBU), placed inside the public buses and granting Internet connectivity to the mobile elements, and RSU, responsible for connecting the mobile elements to the rest of the infrastructure and the Internet. One of the services available in the ATCLL vehicular network is the periodic transmission of status messages, named Cooperative Awareness Messages (CAM), by both the RSUs and OBUs, containing information about the position, speed, and heading, of the sender element. The CAMs transmitted by the vehicles' OBUs and received by the RSUs are collected and stored in the ATCLL datacenter for further analysis, for example, to study mobility profiles. This means that, in similarity to the first dataset, the data corresponds to real trajectories, which is preferable when compared to synthetic datasets that are produced artificially.

The dataset in question is the result of retrieving information about the trajectories performed by all of the 10 buses, during a time frame correspondent to 4 hours. As previously highlighted, this does not produce a rich dataset, but in light of the scenario's complexity, it is sufficient, as proven later in this chapter. Contrarily to Porto's dataset, instead of being provided with a set of trajectories, in this case, we got a set of positioning reports, allowing us to define the trajectories as needed.

Aveiro's dataset was constructed while using a constant time sampling of 1 second, however, this is much higher than the desirable, which would result in repetitive sequences where, most of them, would be characterized by repeating only one RSU. Bearing that in mind, we were required to perform some transformations to the trajectories, re-constructing them in such a manner that, after a location is recorded, its subsequent one becomes the first one observed after 15 seconds or more have passed, discarding all reports observed between them. By opposition, if at any given time no report is observed concerning a specific bus for more than 30 seconds, we assume the trajectory as concluded. The next time that a RSU reports on this bus, a new trajectory is initiated. The process is illustrated in Figure 4.1, where multiple reports concerning the same OBU are received, and then transformed into two separate trajectories. Each report contains the information about when the report is received, the OBU in question, and the RSU that generated the report.

The sequence length considered remains the same that was used for the previous scenario, $\Lambda = 4$, still corresponding to the prediction of only one cell as well. In light of what was enunciated in the last paragraph, this results in sequences that correspond to a

[1] https://aveiro-living-lab.it.pt/

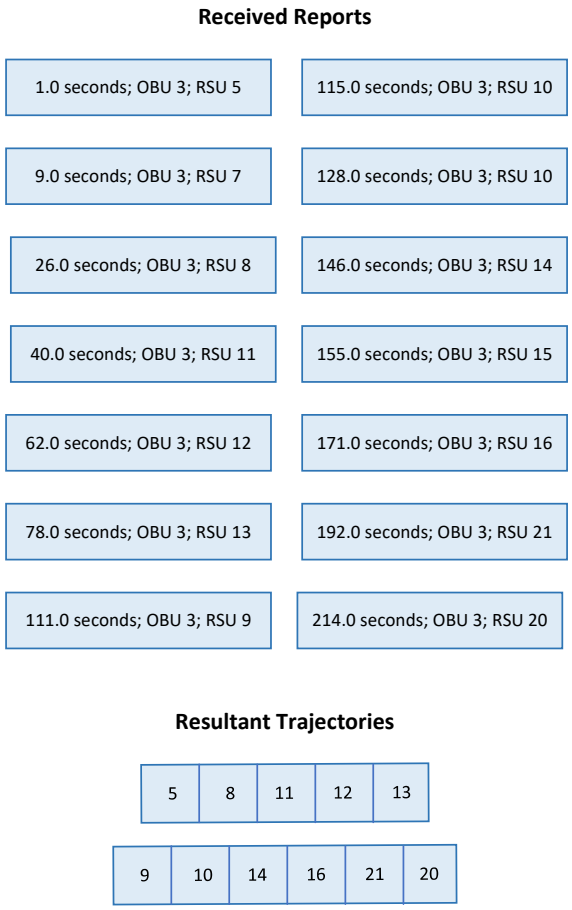


Figure 4.1: Representation of the procedure taken to transform individual reports into trajectories.

time frame somewhere between 45 and 90 seconds.

As stated above, in this scenario there were 21 RSUs in place, which represent the different areas where a bus can be at a given time. Thus, an analogy may be done between the number of existing RSUs and the number of considered cells in the previous scenario, N. However, in this case we do not have squares cells, but an area covered by the RSU antena, which may be affected by physichal obstacles. In short, we may say that the spatial resolution for this second scenario is N = 21, which also means that when performing predictions, there are 21 possible outputs.

When it comes to the segmentation of the data in a training and a testing dataset, the approach taken is similar to the one explained in the first scenario. In this case, of the total of 4 hours of collected data, 75%, corresponding to around 3 hours of data, is used to train the model and, 25%, which represents approximately 1 hour, is used to analyse the predictions' accuracy.

Table 4.1: Analysis of the resultant datasets for both scenarios.

Dataset	Training Dataset		Testing Dataset		
	Total Number of sequences	Unique Sequences	Total Number of Sequences	Unique Sequences	Number of Unknown Sequences
Porto's Data	2 061 326	5 415	691 254	3 382	143 362
Aveiro's Data	47 808	2 079	14 195	1 584	1 782

4.2 Datasets Analytics

In this section we analyse the datasets previously described. The first analysis is at the level of the sequences that result from the partition of every trajectory in overlapping sequences of length $\Lambda = 4$. Recall that the same length is considered for both scenarios.

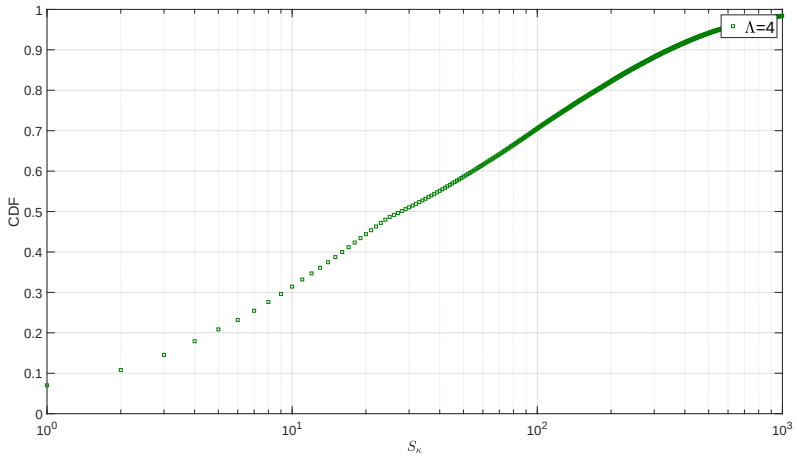
For each scenario, the available data is divided into two datasets, specifically the one used for training the model and the one used while validating it. After performing the division of each trajectory into multiple sequences, we are able to perform a simple initial evaluation of the data in hands for both datasets. We can take a look at the total number of sequences that result from this partition for each of the datasets, as well as the number of unique sequences in each one. Note that, in the training phase all sequences are used, and in the validation phase only the unique sequences are used while considering each unique sequence occurrence probability, which means that this analysis is purely demonstrative. When it comes to the testing dataset another relevant parameter appears, the number of unknown sequences. Since the datasets used in the training and validation phases are not the same, it is probable that some sequences resultant from the testing dataset were not observed during the training phase, which particularly hampers our ability to perform the model's validation. The sequences that were not previously observed are from now on referred to as unknown sequences. The obtained results can be found on Table 4.1.

By performing a quick analysis of Table 4.1, the differences between both datasets become clear. On one hand, both resultant datasets that concern Porto's scenario have abundant amounts of data, where the number of unique sequences is considerably lower than the total number of sequences observed. In opposition, the data resultant from Aveiro is far more scarce, which is demonstrated by having fewer sequences in total and by having considerable high amounts of unique sequences. We emphasize that solely these numbers are not enough to judge the model's capability to accurately perform predictions, as the sequence distribution also has to be taken into account, which will be discussed later.

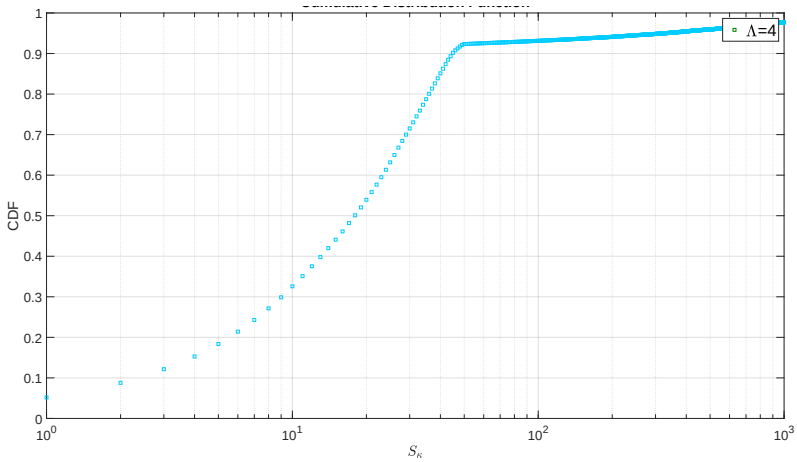
On a different note, we have to also observe the number of unknown sequences. The number of times that unknown sequences were found in the testing phase is significantly higher in Porto's scenario which is not odd since more data is available for this one. However, what is interesting to note is that the percentage of unknown sequences encountered on the testing dataset for this scenario, around 20.74%, is also higher than the percentage found in the data from Aveiro, which is approximately 12.55%. This is also an interesting factor that foresees a balancing of both models' capabilities since this type of sequence

deteriorate our capability to validate the models.

In light of the previous chapter, one factor we still have to consider is the sequence distribution, specifically the distribution of the unique sequences occurrence in the set of all sequences. In other words, we still have to analyze the unique sequences' Cumulative Distribution Function (CDF). Working with more dominant sequences will increase our chances off successfully performing predictions as explained in Section 3.3, since the models rely on the probability of some given sequence to appear. Note that observing the CDFs gives us an idea of the models applicability prior to validating it. The CDFs of the unique sequences are illustrated in Figure 4.2.



(a) Scenario of Porto's taxis.



(b) Scenario of Aveiro's buses.

Figure 4.2: CDFs of the unique sequences for both scenarios.

When analysing Figure 4.2 it becomes evident that there are more dominant sequences in the scenario of Aveiro. Although in both cases 30% of the CDF corresponds to approximately 10 sequences, the difference increases as more sequences are introduced. With that said, in Aveiro, the 100 most probable sequences represent around 93% of the total

of observed sequences, whereas in Porto's scenario the 100 most likely sequences represent only approximately 70% of the total number of observations. This discrepancy was expected since the trajectories taken by buses are far more consistent than the ones performed by taxis.

The traced CDFs demonstrate that the data concerning the Porto's scenario is more representative of the problem and will generate more authentic predictions since there are fewer dominant sequences. On the other hand, for Aveiro, the prediction results will focus mostly only on the top most probable sequences, due to their higher dominance. This also shows that, despite the scarce amount of data concerning Aveiro, the model is probably more than capable to perform accurate predictions because of the dominance of some sequences over others, probably even outperforming Porto's model accuracy. This again may be explained by the high consistency of buses' trajectories.

4.3 Mobility Prediction Evaluation

In this section, we explain how the models' capabilities to perform predictions were validated, and present their accuracy results. The methodology employed here is similar to the one defined in Section 3.2, so the reader may refer to that chapter's section for a deeper understanding of the principles behind the validation.

As enunciated before, the sequence length is the same for both scenarios, $\Lambda = 4$, however, the spatial resolution is different. For the scenario of Porto, a resolution of $N = 64$ was chosen, and in Aveiro, the infrastructure is composed by 21 RSUs which results in having a spatial resolution of $N = 21$. What is different as well, is the type of algorithms used in each scenario, which are going to be described next.

For the scenario of Porto, we opted to use the ANN based approach, since it achieved approximately the same prediction accuracy as the LSTM based one, while still utilizing fewer computational resources, resulting in less computational time. The constructed model is similar to the one presented in Section 3.2, however, this time is composed of 3 sequential dense layers. We opted to use only 3 layers this time since the amount of data to consider while tuning the model's parameters is considerably lower. All of these utilize ReLU as their activation function, apart from the last one that needs to limit the model's output between [0,1], therefore using $softmax$ as its activation function. The model was trained while using the categorical cross-entropy loss function, during 100 epochs with a batch size of 512 and using the Adam optimizer. The utilized configuration can be visualized in Table 4.2.

The Aveiro model uses an LSTM based approach in an effort to counterbalance the lack of data and to achieve the best possible results with the available data, since it is considered to usually achieve better performance results than the ANN one. This is done despite the fact that we were able to achieve similar results with both approaches, for the Porto's scenario. The model comprises two sequential layers, the first one being an LSTM layer composed of 16 hidden units that use $tanh$ as their activation function. The

Table 4.2: Configuration of the ANN model' layers used for Porto's scenario when only 75% of the dataset is used for training, adapted from [17].

Activation Function		Employed Loss Function	Utilized Optimizer	Number of Epochs	Batch Size
Initial Layers	Final Layer				
ReLu	Softmax	Mean squared error	Adam	100	512

Table 4.3: Configuration of the LSTM model utilized in Aveiro, adapted from [17].

LSTM Layer	Activation Function		Employed Loss Function	Utilized Optimizer	Number of Epochs	Batch Size
	LSTM Layer	Dense Layer				
16 hidden units	Tanh	Sigmoid	Mean squared error	Adam	100	512

second layer is a dense one that has a *sigmoid* as its activation function. During the training phase, an Adam optimizer was utilized, using the same number of epochs, 100, and batch size, 512. In this case, the loss function employed was the MSE. The described configuration is summed up in Table 4.3.

Similar to what was explained in the previous chapter, the models' inputs take different shapes. In the case of the Aveiro's LSTM based approach, the model's expected input is a matrix with shape $\alpha \times N$, which in this case, translates into a 3×21 matrix. For Porto's ANN model, the input is a one-dimensional vector of $\alpha \times N$ columns, or in order words, a vector composed of 3×64 columns. However, both models' output shape is the same, a vector of $(\Lambda - \alpha) \times N$, which actually translates into a vector of N columns.

The method to study the prediction accuracy is also similar to the one employed in the previous chapter. A prediction is performed for each unique sequence present on the testing dataset. Afterwards, the prediction result, 1 if correctly predicted or 0 otherwise, will be multiplied by the occurrence probability of the sequence in question. By summing all of the calculated results, we will obtain the accuracy, or in other words, the PP. Note that these occurrence probabilities are obtained from the training dataset, therefore only representing an approximation of their probable occurrence on the testing dataset.

The biggest difference in the accuracy calculation between this chapter and the previous one is the introduction of unknown sequences. Validating a model's accuracy becomes particularly harder, since there is no occurrence probability associated with a sequence that was not observed in the training phase. Therefore, the PP obtained before is not realistic, forcing us to tweak the accuracy calculation.

To obtain a more reliable value we must firstly calculate the accuracy as explained in the previous chapter by

$$accuracy = 100 \sum_{i=\alpha+1}^{\Lambda} p_{S_m} F(c^i_{pred}, c^i_{corr}), \tag{4.1}$$

where F is the function that compares the cell predicted by the model, c^i_{pred}, and the

corresponding correct cell, c^i_{corr}, and returns 1 if prediction is correct, or 0 if not. Afterwards the value returned by the function is multiplied by the unique sequence occurrence probability in the data, p_{S_m}. Then we are able to determine the number of correct predictions, $n_{correctpredictions}$, made as long as we have information about the total number of sequences, $n_{totalpredictions}$, in the testing data set as well as the number of observed unknown sequences, $n_{unknownsequences}$, as

$$n_{correctpredictions} = \frac{accuracy * (n_{totalpredictions} - n_{unknownsequences})}{100}, \tag{4.2}$$

which allows us to define the PP as

$$PP = \frac{n_{correctpredictions}}{n_{totalpredictions}} * 100, \tag{4.3}$$

therefore obtaining a more realistic PP value.

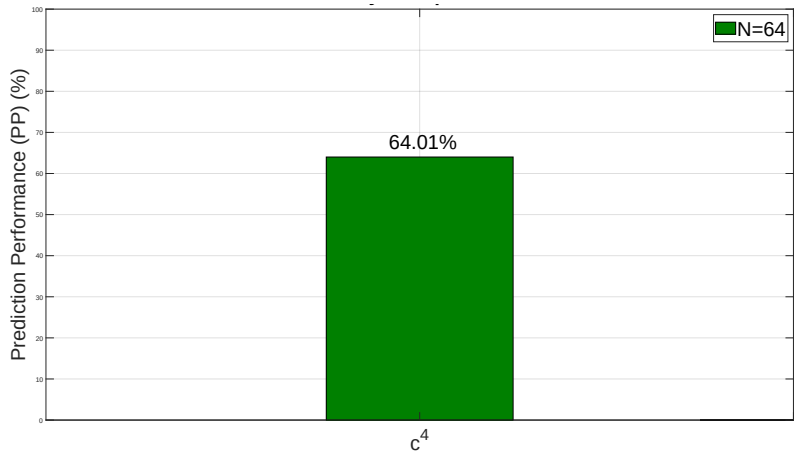
Using (4.3), we are able to plot the prediction accuracy obtained for both scenarios, as seen in Figure 4.3. From this, it is possible to observe that the accuracy obtained for the scenario in Aveiro is much higher than the one in Porto. This disparity happens in despite of the lack of data collected from Aveiro, mostly due to the homogeneous trajectories described by buses, whereas the trajectories collected from taxis are much more irregular. Besides, the spatial sampling also decreases for the scenario concerning Aveiro's buses, which also foresees an increase in the accuracy as demonstrated in Section 3.4.

We should reinforce that the plotted graphs do not correspond merely to the percentage of times that predictions were done correctly. The accuracy values may seem lower than expected when compared to the ones obtained in the previous chapter. However, this is easily explained not only by the shrinkage of the available data during the training phase, but also by the introduction of unknown sequences that are always considered as incorrect predictions. In fact, prior to taking into account the unknown sequences we were obtaining PP values of 80.76% and 94.30% for Porto's and Aveiro's scenarios respectively. These values dropped to 64.01% and 82.74% respectively, which clearly demonstrates the impact that unknown sequences have in our validation process.

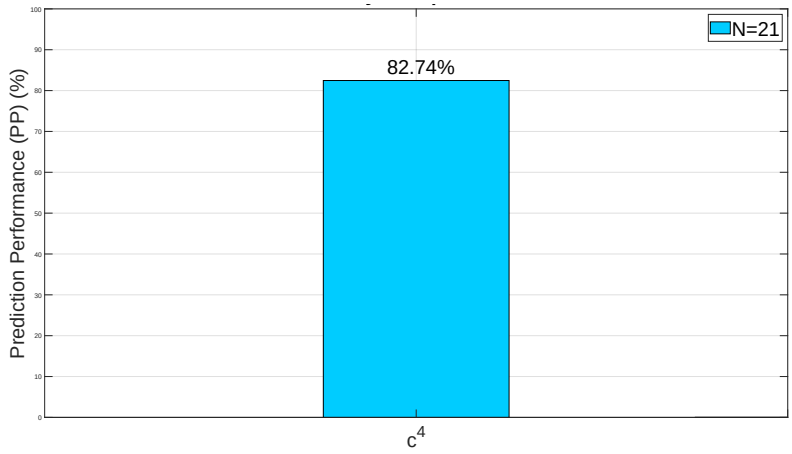
4.4 Network Improvements

The last point of this chapter focuses on demonstrating how the mobility prediction results presented in the previous section could be used to improve, for example, the performance of a Content Distribution Network (CDN) supported by a vehicular network. The idea is to explore the pre-caching mechanism, as explained in Section 2.3, where a given content is stored, in advance, closer to the end-user for future use.

In a system without pre-caching mechanism, when a user wishes to access any type of content, he/she needs to be able to reach a specific server which contains the data to download it in that specific instant. This is not only time-consuming but also provokes



(a) Scenario of Porto's taxis.



(b) Scenario of Aveiro's buses.

Figure 4.3: Prediction Performance for both scenarios.

bad management of the network resources. Pre-caching is an attempt to resolve this issue and, while not being perfect, it may bring considerable enhancements to the network management. In other words, if a given Content originally in the Cloud is moved to the RSU in advance, the retrieval time is reduced.

We propose to use mobility prediction to enable the implementation of a pre-caching mechanism for a system where the end-user is located inside the vehicle. The mobility prediction is performed based on the moving entity's last known locations, and the next chunk of content may be stored in advance on the most probable future user's location. From then, there are two possible outcomes:

- The mobility prediction is incorrect. This means that the next chunk of data to be requested by the end-user will not be stored in the next visiting RSU. Therefore, when the user requests the chunk of data from the RSU, the data will not be available there, requiring another download of the same data from the Cloud again.
- The prediction about the entity's next location is correct, and therefore, the chunk

of data to be requested by the end-user will be stored in the next visiting RSU, and can be downloaded from there. This result in a shorter retrieval time since the data is sent directly from the RSU to the end-user.

The reliability of such a system depends heavily on the model's capability of performing accurate predictions. In case the model is absolutely not capable of performing such predictions accurately, the user will need to download the data from the server again, resulting in increased network overhead, since those chunks of data were already downloaded, and in poor management of resources at the RSUs since they are required to store this information for a certain period. On the other hand, if the model's predictions are flawless, the data will be available at the right RSU at almost all times, resulting in faster downloads from the user's point of view.

Another benefit provided by such a system is not at the level of data delivery time but in resource management. A network management system can leverage the pre-caching mechanism to download data in moments where the network traffic is lower, and therefore take advantage of its free resources when these are not in use.

To understand the benefits of the pre-caching mechanism in the mobility scenarios considered in this work we first need to measure the two-way delay, also denoted as Round Trip Time (RTT) between the OBU and RSU, and between the OBU and the Cloud. To that end, and using the vehicular network of the ATCLL we configured an OBU placed inside a public bus to periodically and simultaneously perform *pings* to the connected RSU and with a server placed in the United States of America (acting as the Cloud). The ping tool was configured to use ICMP messages (ICMP echo requests and ICMP echo replies) of size 64 bytes. Excluding the *ping* events without RTT records, due to packet loss events, we were able to collect 4650 RTT measures between the OBU and the RSU, and 4086 RTT measures between the OBU and the Cloud.

As expected, the RTT to the Cloud is usually higher than the one to the RSU. This may be observed in Figure 4.4, where a representative portion of the histograms for both sets of pings may be seen.

In an effort to show how beneficial this system could be we plotted and compared the CDFs of the RTT when three scenarios are considered: the first one admits that all the communication occurs between the OBU and the RSU; the second one admits that all the communications occurs between the OBU and the Cloud; and a third one assuming that the communication occurs between the RSU and the Cloud with a given probability. By other words, and assuming a scenario of pre-caching CDN, the first scenario considers that the Content requested by the end-user is correctly pre-cached in the visiting RSU, the second scenario assumes that the Content is only stored in the Cloud while in the third scenario the Content is pre-cached in the RSU with a PP given by Figure 4.3. To plot the last one, we analysed how many of the total of predictions were performed correctly, and how many were incorrect. Afterwards, for each correct prediction, we executed an inverse transform sampling from the CDF of the RTTs to the RSU. After repeating the

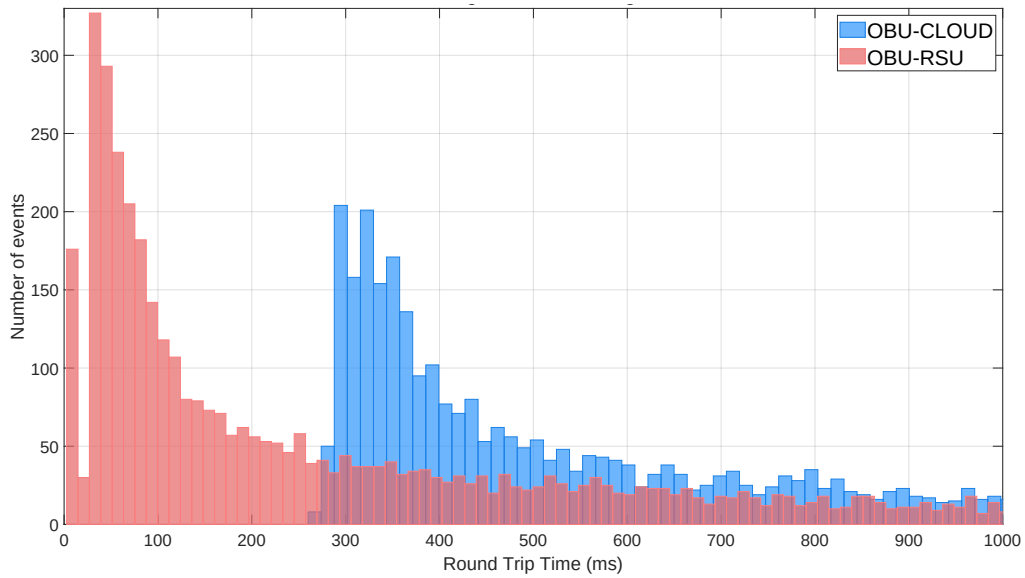


Figure 4.4: Histogram of the RTT of multiple *pings* sent to both the RSU and to the Cloud.

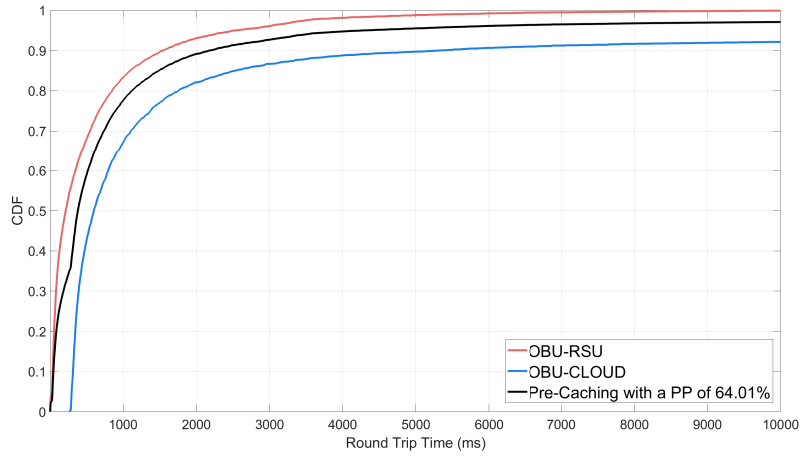
same process for each incorrect prediction, now using the CDF of the RTTs to the Cloud, we agglomerated all the obtained values and traced the desired CDF.

The process described above was repeated for both scenarios, in Porto and in Aveiro, resulting in the two CDFs presented in Figure 4.5. In each sub-figure, it is possible to observe the three earlier mentioned CDFs, where the red one represents a perfect scenario where all the predictions made are correct, the blue one represents exactly the opposite when all predictions are incorrect, and lastly, the black one represents what we should expect to obtain if a pre-caching mechanism was implemented that used the prediction models presented in the previous sections.

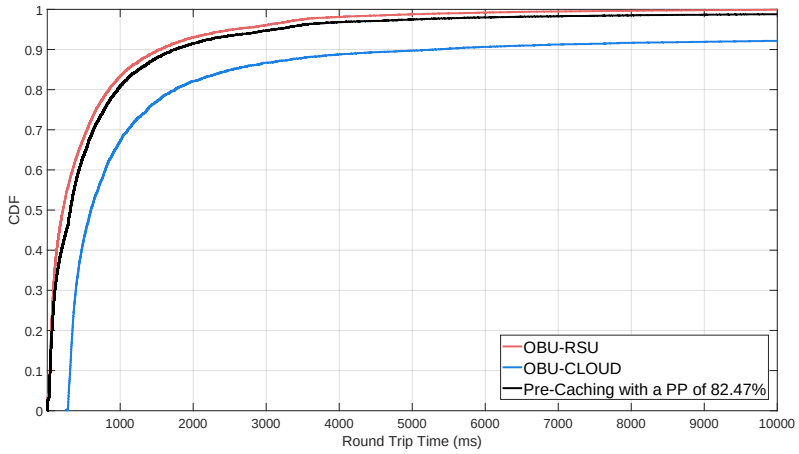
From Figure 4.5, it is possible to see that the CDF plotted for the case when the pre-caching is implemented, sits between the other two CDFs, which is expected since these are the optimal and worst-case scenarios. It is also possible to observe that for the Aveiro scenario, Figure 4.5(b), the CDF is much closer to the optimal one than in Porto, Figure 4.5(a). This happens because, as seen in Section 4.3, the prediction model for Aveiro has demonstrated a higher prediction accuracy than the model created for Porto. This will result in more RTTs taken from the optimal CDF, which will then create a more preferable CDF.

One important note to make is that the illustrated CDFs correspond to the RTTs from the packets sent. Although, our intention is not to study the two-way travel time, but the data´s retrieval time which will be different since the data download and its control are also performed. However, our study is still applicable because the introduced delay would be the same throughout the different cases. In practical terms, this means that if we were tracing the CDFs of actual retrieval times instead, it would be possible to observe the same improvements in terms of time, as observable on the traced graphic.

We should highlight the fact that what we refer to as the worst-case scenario, when



(a) Scenario of Porto's taxis.



(b) Scenario of Aveiro's buses.

Figure 4.5: Representation of how the CDF of the RTTs would look for each scenario if a pre-caching mechanism was implemented, considering its PP illustrated in Figure 4.3.

all RTT values are taken from the communication with the Cloud, is actually the current behaviour of any system that does not thrive to implement a similar system to the proposed one. By analysing Figure 4.5, the benefits of implementing it become evident as, the discrepancy between the purple and the blue CDFs is considerable, even in a scenario where only 64.01% of the predictions are correct. On the other hand, if implemented in a scenario with 82.47% mobility prediction accuracy, which is the same as saying it incorrectly estimates almost 17.53% of the predictions, the obtained results will still be much identical to the ones in the optimal scenario. This clearly demonstrates the benefits of implementing a pre-caching mechanism in networks that expect mobile users, and anticipates the magnitude of the improvements that should be expected for similar scenarios.

4.5 Summary

To summarize, in this chapter, we started by presenting and studying two different mobility prediction scenarios, concerning the cities of Porto and Aveiro. We explained what both scenarios entailed, and performed a pre-prediction study of the data available, and consequent resultant sequences, for the respective datasets. Afterwards, we described the methodology used to perform the mobility predictions, as well as the techniques employed to validate them. To finalize, we described how mobility prediction could be used to create a pre-caching mechanism on a CDN, in order to achieve some improvement on the communications network management.

An important remark to do is that this study was performed in a theoretical way. If such mechanism was implemented in a telecommunications network, the actual enhancements achieved should be lower than the ones demonstrated by us. The main explanation to this conclusion is that, by only performing the study theoretically, we have no way to study the impact that the system's deployment would take on the network. An example is the overhead introduced by the mechanism's control messages, which would certainly increase the resource utilization, and perhaps the network's average delay. The impact would be even heavier in scenarios where the mobility prediction presents lower accuracy results due to the repetition of the Content's download from the Cloud. The next chapter will focus on the implementation of a prototype responsible for performing the mobility prediction.

5

Aveiro's Prototype for Mobility Prediction

In the previous chapters, we described how we perform mobility prediction utilizing both LSTM and ANN approaches on different scenarios, and demonstrated how utilizing these predictions to implement a pre-caching mechanism would improve network management. In this chapter, we describe the mobility prediction prototype developed to take advantage of the infrastructures already in place in the city of Aveiro, which retrieves information about the mobility of public buses that are equipped with the appropriate communication equipment.

Similarly to what was presented for some previous models, we also implemented an LSTM based model to perform the mobility prediction in the city of Aveiro. However, we also developed an application that has the ability to interact with a third party application, performing the predictions as requested. Beyond this addition, a major difference between the implementation made in the previous chapters and the one to be presented next is that whereas in the previous ones we utilized a certain dataset, now we intended to receive the data concerning the buses' mobility in real time, gather it in a centralized manner, and finally use it to construct our datasets.

We initiated this chapter by explaining in Section 5.1 the structure of the prototype, as well as, explaining how we handled the problem. Afterwards, in Section 5.2, we describe the different moments not only in the construction of the prototype but also in its functioning. In Section 5.3, we present the model developed to perform the predictions, and analyze its PP results on Section 5.4. Lastly, we present a few final notes on the prototype and its interaction through the console in Section 5.5.

5.1 Prototype Architecture

The prototype to be presented next is implemented in such manner that it will continuously make predictions on a request basis, which is essential when one wants to integrate these mobility predictions capabilities in a pre-caching mechanism. The idea behind it, is separating the mechanism into two parts, where the first is a pre-caching algorithm that is aware of the user's observable locations and online traffic, which would

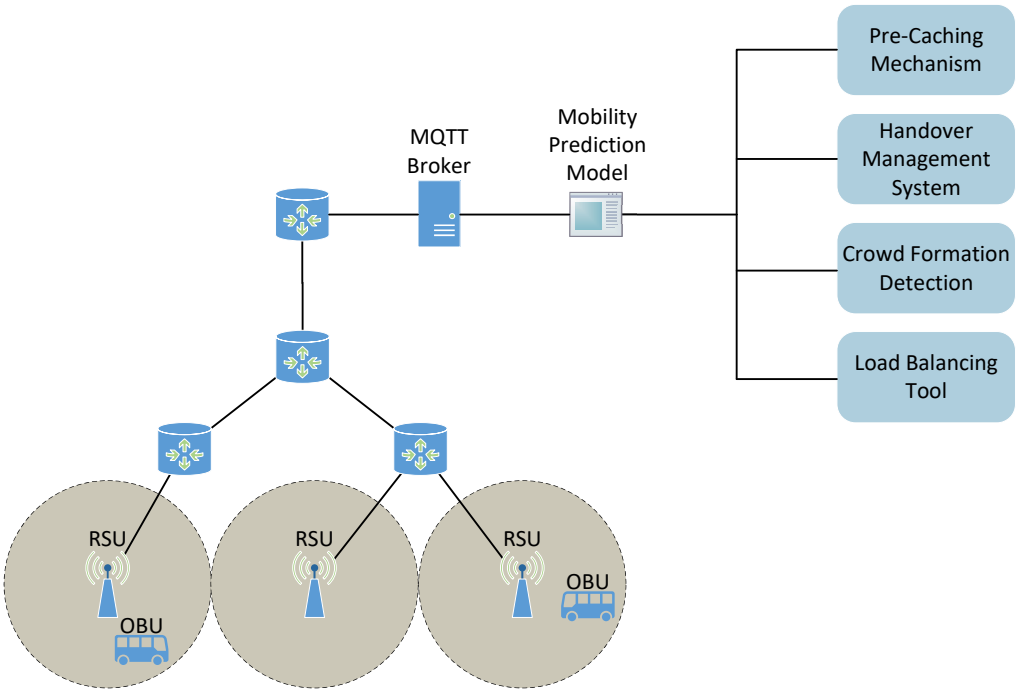


Figure 5.1: High-level overall view of the structure surrounding the mobility prediction prototype.

then determine how much data and which data to download. Then, it would pass the user's known previous locations to the second part, the mobility prediction model, so that this one could respond with the most probable future location, allowing the pre-caching mechanism to store the information in the correct RSU. Our work concentrates only on developing the real-time mobility predictor.

As mentioned, the prototype was developed while taking into consideration the infrastructure in Aveiro. The specific structure which assures the flow of information from the buses until the machine where the mobility estimations are performed, will be explained in the next section. However, if someone would intended to replicate a similar application in another city, there would be no need for the structure to be exactly the same, as long as the connectivity between the mentioned endpoints exists.

One important remark to do is that, although we developed our prototype aiming at the implementation of a pre-caching mechanism, it is not in anyway limited to this. The prototype can be implemented to support different services, such as enhancing handover mechanisms in cellular networks, among many others. Some different services that could avail such prototype are represented in Figure 5.1, as well as representation of a portion of the infrastructure in use.

In this way, it is clear that the proposed prototype, or an adaptation of it, can bring improvements to different areas. Such modifications, can also be implemented in the core structure of the prototype itself as one sees fit. The developed prototype can be divided into three main components which are described in the next points, and represented in Figure 5.2:

(a) The first one, denominated as the data gathering module, is responsible for receiving any announced report and processing the information. However, in our case, it does not store every received report since it also is the module that composes the trajectories. Similarly to what was explained in Section 4.1, in this case, we were confronted again with a very high sampling rate of 1 second. Due to this we utilized the same approach to decrease it by discarding any report received within 15 seconds from the last one considered for the trajectory formation. Again, it is also necessary to limit the time frame during which we wait for another report, 30 seconds, after which the trajectory is considered finalized. As an output, this module produces daily files in .csv format, where each one corresponds to a dataset concerning the information obtained during that day. Beyond these, it is constantly updating another .csv file with the information about which RSUs have already reported. The gathering module is running indefinitely in a forever loop, always gathering more information and updating the correspondent files;

(b) The output files from the previous module are the input of the model training one. It is not necessary to use all the daily datasets, in fact, it may not be advisable when the available data is abundant. Therefore, this module input will only be a determined number of files, typically, the most recent ones. With that data, the module is capable of transforming the trajectories into sequences which will then be used in the model's training process. This module outputs a file with .h5 extension, that contains the fully trained model ready to perform the necessary predictions, which is used in the last module, as well as in the model's offline validation if one should want to perform it. To keep the prototype up-to-date the model's training is repeated from scratch every day at 00:00, with the most recent information concerning trajectories and the RSUs that have already reported;

(c) The last module is responsible for achieving the prototype's ultimate goal. As said above, this module takes the trained model as an input and then communicates with another application, waiting for data concerning an entity's last known locations. On receiving it, the module will utilize the trained model to perform the prediction and, it will reply to the other application with the result of that same prediction which will then use this information. As the first module, this one must be kept running indefinitely, always responding to the applications served by our prototype. The newly trained model is imported at 01:00, which assumes that the training phase will not take longer than 1 hour, although this time may require modification for more complex scenarios.

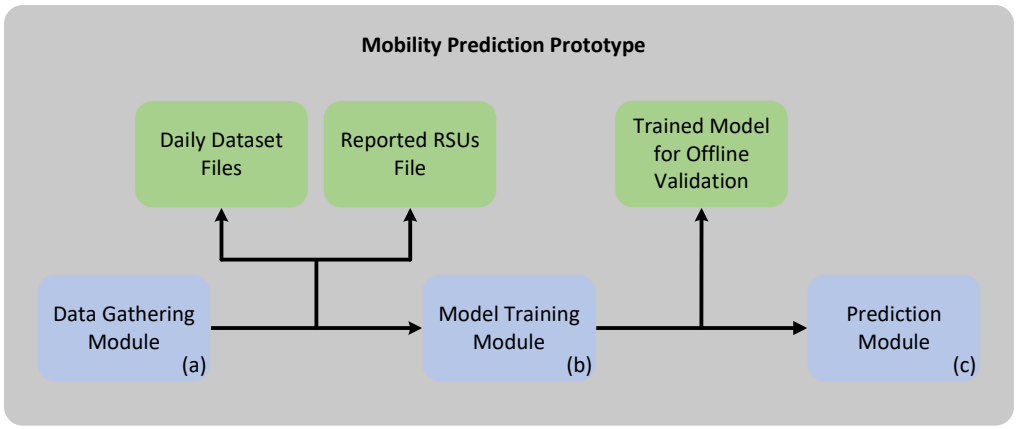


Figure 5.2: Structural representation of the prototype.

5.2 Workflow

From now on, we are going to take particular focus on how we handled the problem and the infrastructure we were confronted with, however, the reader should still have in mind that the presented work can be tweaked to support any similar applications.

The implementation of such prototype consists on multiple phases that come sequentially, as it is possible to observe in Figure 5.3. As stated before, the mobility information utilized in this chapter does not consist on a pre-prepared dataset but rather on data gathered in real-time. Intuitively, this should be the first step, which is explained next. By continuously acquiring data, and assuming that none is being deleted, at some point, there will be more data available than the desirable to use in the model's training phase if we consider that more data translates into more training time. Therefore, the second phase is determining the amount of data that should be used to train the model, which should vary according to the scenario and its complexity. The third phase is determining which sequence length to use. Afterwards, we are capable of performing the transformation of the acquired data into trajectories into sequences composed of Λ cells, as determined in the previous step. The fifth phase is the training of the model itself, which has an approach similar to the ones discussed in the previous chapters. As mentioned above, this step is repeated every day, at 00:00, using the most recent data. After the training is complete, a python script imports the model capable of performing predictions, which, while running in a loop, will be continuously expecting that the other application requests predictions, outputting the result back to the application when so. Similarly to the previous module, this one will import the trained model every day at 01:00.

As stated previously, the data described in this section refers to data received in real-time that concerns a set of buses circulating in the city of Aveiro. The next paragraphs aim at disclosing how the data acquisition has taken place, which could be maintained if more vehicles or RSUs were considered, or even if applied to different scenarios in different cities.

As previously introduced, various city buses were equipped with a communication

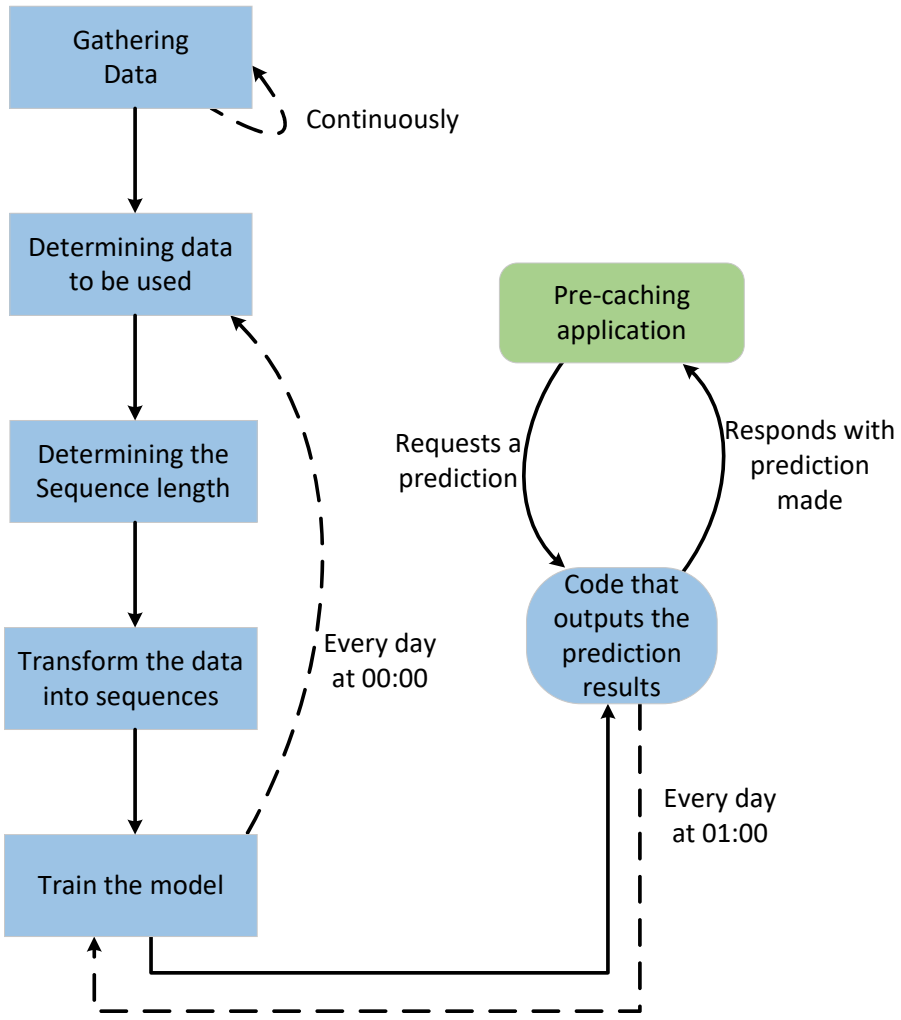


Figure 5.3: Phases that compose the prototype's functioning.

device, referred to as OBU, that allows them to communicate over short distances. On the other hand, there are also some RSUs placed in strategic places along the roads where the buses are expected to pass through. These RSUs work as the receivers of the buses' announcements. It is crucial to understand that each bus is identified by its unique OBU Media Access Control (MAC) address, and each RSU is distinguished by its RSU ID, making it possible to determine precisely which bus is reporting to exactly which RSU, allowing us to know the whereabouts of the bus since the location of each RSU is known and constant. Ultimately, receiving consecutive reports about one OBU enables the creation of the trajectories, and subsequent sequences used in the model's training phase.

There are numerous deployed RSUs throughout the considered area, more specifically 21 with prospects of increasing this number in a close future. These units send reports every 1 second to a centralized controller via TCP, concerning the buses present in its range, if any, at the time. Therefore the information regarding all the RSUs is agglomerated in one spinal node that is reachable by any stationary unit. After receiving the

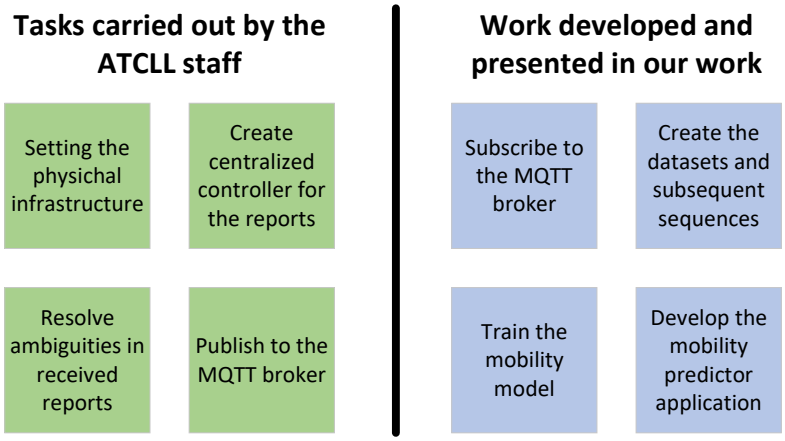


Figure 5.4: Separation of duties between the authors and the ATCLL staff.

various reports, the centralized controller resolves any ambiguity present. Although not frequent, these ambiguities take place when an OBU is in the range of two or more RSUs, resulting in having reports from different stationary units concerning the same bus. In this case, the controller elects one report as the better one by comparing the different signal strengths and choosing the higher quality one. Only then, the data is ready to be utilized.

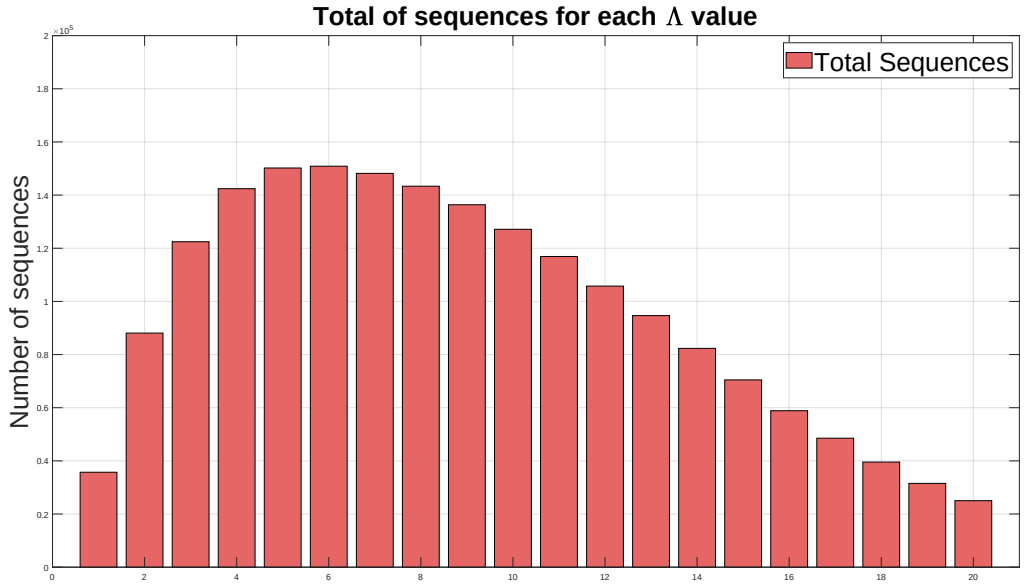
To retrieve this data, we opted for it to be passed from the centralized controller to the virtual machine where our mobility prediction is performed via a Message Queuing Telemetry Transport (MQTT) server. MQTT is a publish/subscribe technology commonly found in Internet of Things (IoT) infrastructures, mainly due to the fact that it creates infrastructures easily expandable. As explained by the authors in [15], an MQTT broker is created as the backbone of the technology, broker to which the entities that will generate the information, the centralized controller in our case, must register as publishers. Afterwards, any entity that wishes to receive the generated information, as the code in our virtual machine, must register as a subscriber. As stated, this technology is very scalable since any other entity, publisher or subscriber, may be registered at any given time without disrupting the current adjacencies.

It is crucial to emphasize that the deployment of described infrastructure, and consequent retrieved information, was performed by the ATCLL staff, who made it available to our virtual machine, responsible for the mobility prediction. With that said, all tasks performed before the publication of the information in the MQTT broker were performed by the ATCLL staff, and all subsequent ones by us. Figure 5.4 emphasizes this separation of tasks.

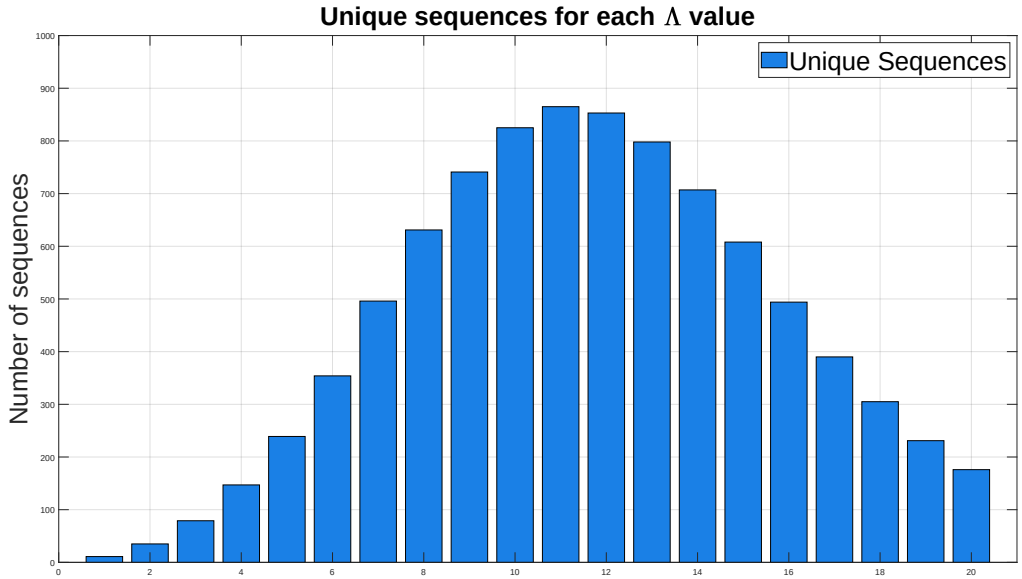
As explained in Section 5.1, the data gathering module continuously saves the information received to files that work as our dataset, where each file correlates to the date in question. It consists on a Python script, that subscribes to the MQTT broker and then treats the data as explained. The fact that the data is separated through files allows us to

experiment with different configurations using, for example, the data received throughout 1, 3 or 7 days.

In light of the study performed for previous scenarios, with the data now acquired from Aveiro we also intended to perform predictions for different Λ values. Therefore, we felt the need to perform the partition accordingly to these values and study the resultant number of sequences, number of unique sequences and their cumulative distribution. However, since the dataset is not static anymore, this research is not as straightforward as before.



(a) Total number of sequences.



(b) Number of unique sequences.

Figure 5.5: The resulting number of sequences and unique sequences, when the information retrieved along 7 days was partitioned with $\Lambda = \{1, 2, 3, ..., 20\}$.

Before starting the prototype's development, we had to define how many files we wished to train the model with, along with the Λ value and the respective number of cells that should be predicted when constructing it. Therefore we plotted two histograms, one concerning the total number of sequences and the other only the unique ones, where each instance correlates to a different Λ value. We plotted and analyzed both histograms using from 1 to only 8 daily files. In Figure 5.5, it is possible to see an example of the histograms taken, in this case using 7 files, or in other words, the information retrieved along 7 full days. Note that more histograms were taken but will not be presented here due to their similarity.

Through analysis of the various histograms plotted and the corresponding numbers of sequences for each Λ value, we opted to study the prediction performance for $\Lambda = \{4, 8\}$ predicting respectively 1 and 3 future cells. Note that this work was not done in a real-time manner, serving only the purpose of validating the model and a further selection of the best sequence length to use in the prototype.

Once the sequence lengths to consider were chosen, we were able to study the cumulative distribution of the unique sequences, in order to understand if we should expect to have very dominant sequences, or not, according to the number of data files utilized. For this analysis, one should focus on the results presented in Figure 5.6, where 5.6(a) corresponds to the distribution observed when the sequence partition is performed considering $\Lambda = 4$ and 5.6(b) to when $\Lambda = 8$ is utilized.

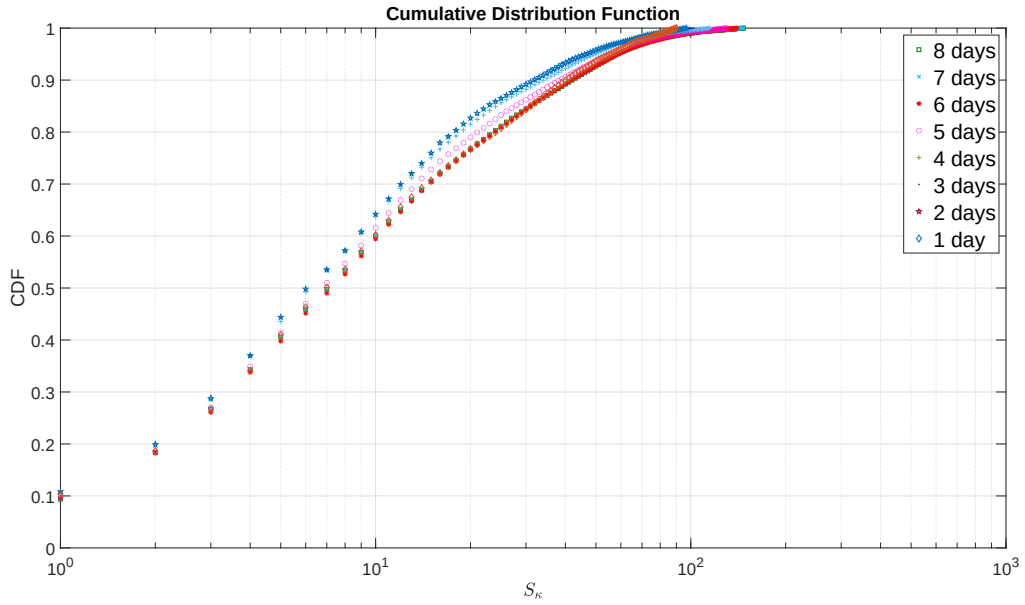
By analysing the results from the cumulative distribution functions, it becomes evident that the number of files does not seem to have a heavy impact, which may be indicative that using the information retrieved from only one day can be sufficient to characterize satisfactorily the problem at hand. The following sections support this conclusion.

On the other hand, we can verify that when the partition is performed considering longer sequences, we encounter less dominant sequences. It is observable that for short-term predictions, the 10 more probable sequences correspond to approximately 60% of the sequences occurrence and, for long-term, the same 10 sequences only correspond to approximately 20%.

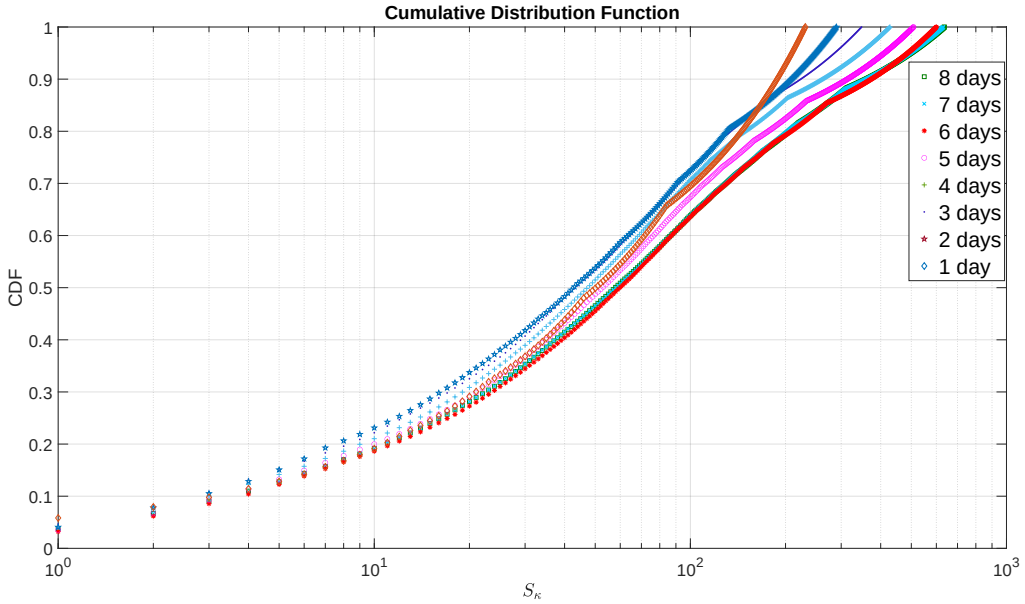
5.3 Training Module

Note that the methodology utilized to perform the models' validations is very similar to the one described in Chapter 3, so, for deeper understanding, the reader may relate to Section 3.2 specifically.

As previously stated, the LSTM based models cope very well with the vanishing gradient problem and consequently are very prone to achieve satisfactory results in scenarios where the model is required to have good long-term memory. For this reason, we opted to perform the development of a LSTM based model, which is expected to outperform the alternative approaches especially when predicting more than 1 future cell.



(a) Cumulative distribution for short-term predictions.



(b) Cumulative distribution for long-term predictions.

Figure 5.6: Cumulative distribution of the unique sequences, according to the different number of information files utilized.

Contrarily to what was do done for the previous scenarios, we limit the characterization to $\Lambda = \{4, 8\}$, which relates to the prediction of only 1 and 3 cells, respectively. This chapter ultimate goal is the creation of a prototype, and having that in mind, using higher numbers of predicted cells is a disadvantage. To make accurate predictions concerning more cells, more information about past cells is necessary, and this supports the previous statement since the prototype will require more idle time before performing its first prediction. Besides, requiring a longer idle time before being able to predict, the models' training phase will also take longer, which may not be desirable for real-time applications.

It is important to note that as we receive reports originated by different RSUs, we are required to save the information about which ones have already sent at least one report. This information is indispensable due to the one-hot encoding. Imagine a scenario where only two RSUs have reported yet, one having the RSU ID 1 and the other an RSU ID 19. To perform the binary encoding as it is, it would be required to use a matrix with 19 columns. To resolve this inefficiency, we opted to translate the RSU ID into a so called cell ID, meaning that we created a vector with the IDs of all the RSUs that have already reported, and identify the RSU according to its index value, or in other words, its cell ID. This vector is kept at all times in order to perform the translation in the inverse direction at the end of the prediction to allow us to determine which RSU is the most probable one. This means that, although in this scenario an entity has 19 possible states, as far as the model is concerned, the spatial resolution is equal to the number of RSUs that have already reported. Not only this optimizes the process as described, it also allows the infrastructure to expand without affecting the prototype.

The prediction model is made of two layers, the first of which is an LSTM layer, and the second a dense layer. The primary LSTM layer is made of 16 hidden units, each of which uses the *tanh* as activation function. The *sigmoid* is employed as the activation function for the dense layer. When it comes to the fitting phase, we opted to maintain the use of the Adam optimizer, while using the MSE as the loss function since we achieved better results with it. A small difference in this phase was that, despite using 1000 epochs with a batch size of 512 samples to train the model when $\Lambda = 4$, this setting where insufficient to fit the model in a satisfactory manner for $\Lambda = 8$, which forced us to use 2000 epochs in the second case, while maintaining the same batch size. The disparity happens mainly due to the lack of data, which forces as to employ a deeper and heavier learning process that in previous scenarios.

In terms of inputs, the model expects to receive a matrix of shape $\alpha \times N$, where α is the number of previously observed cells and N relates to the total number of RSUs that have already reported and, therefore need to be taken into consideration due to the necessity to perform one-hot encoding. On a similar note, the output returned by the model has a shape of $(\Lambda - \alpha) \times N$ where $(\Lambda - \alpha)$ is the sequence length minus the previous cells, or in other words, the number of future predicted ones.

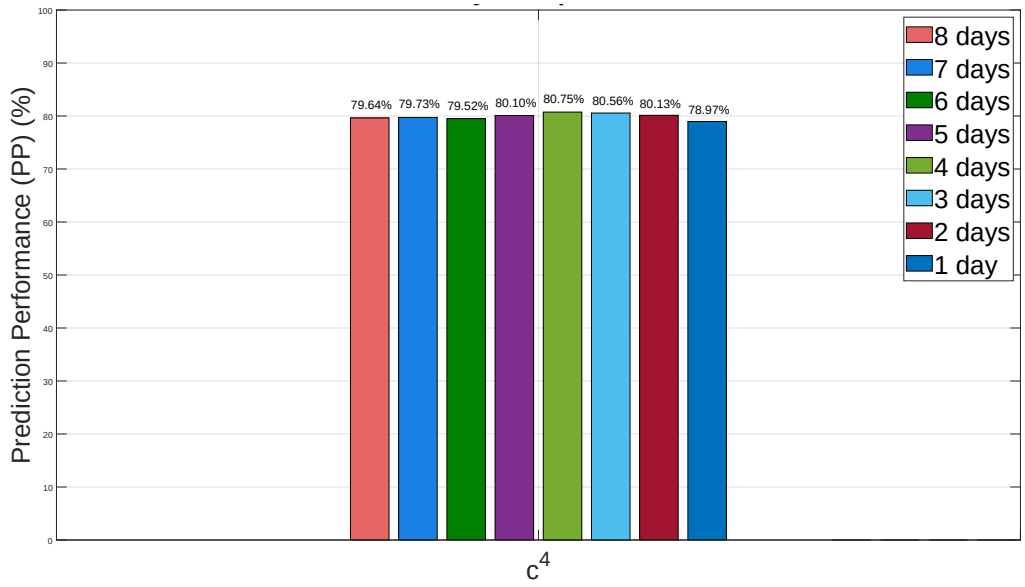


Figure 5.7: Prediction performance results for Aveiro when performing short-term predictions, according to the number of files (days) utilized to train the model.

5.4 Prediction Module

The last step to take before the implementation of the prototype is the analysis of the obtained results in terms of prediction performance to validate the capacity of the model to perform such predictions in the described scenario. As explained above, we opted to perform the predictions while considering only $\Lambda = \{4, 8\}$, and in this section we will study the PP and associated metrics achieved in both scenarios.

The prediction performance results for when the three previous cells are used to perform short-term predictions, may be observed in Figure 5.7. Note that it presents 8 different instances, where each one differs in the amount of data files used to train the model. Remember that one file is populated daily, so saying that 7 files were used to train the model, is the same as saying that the data used to train it, was retrieved over the course of 7 days.

By making a quick observation of the bars chart, is clear that the results were very stable independently of the number of files utilized to train the model. The chart shows almost no loss in terms of accuracy when only 1 file is considered or, in other words, demonstrates that the information acquired throughout one day should be sufficient to train a model capable of performing satisfactory predictions. Therefore, it exhibits the necessity of low amounts of data in order to train the model satisfactorily, in scenarios as simple as this one.

On the other hand, a few more conclusions can be taken from Figure 5.8, which relates to the the prediction performance results obtained when a sequence length of $\Lambda = 8$ is considered. Similarly to the previous figure, each predicted cell has 8 different instances which, again, correlate to the number of days over which the considered data was gathered.

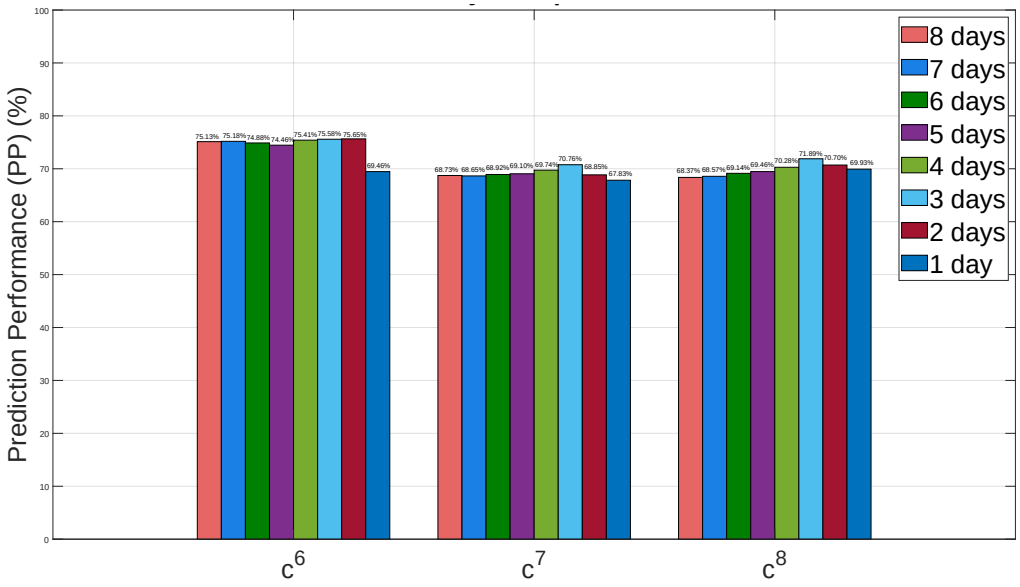


Figure 5.8: Prediction performance results for Aveiro when performing long-term predictions, accordingly to the number of files (days) utilized to train the model.

From the obtained results, one evident conclusion when compared with the short-term prediction ones, is that the accuracy of the first predicted cell is lesser when $\Lambda = 8$ is considered. At a first glance, and in light of what was already explained in previous chapters, these results may be unexpected since we are using more history to predict one cell, however, they are justified by the partition and distribution of the considered sequences studied before. When analysing Figure 5.5, it is possible to observe that with the increase of the sequence length from 4 to 8 cells, the resulting number of unique sequences quadruplicates while the total number of sequences stays approximately constant. This means that when we are validating the model, we have the same number of predictions to perform but with a lot more different sequences as possible states, which justifies the increase of wrongful predictions. Another indicator of the performance decrease is given by Figure 5.6 where it is possible to observe that the unique sequences become less dominant as we increase the sequence length. Since the dominant sequences are the ones most likely to be appointed by the model prediction, increasing the number of possible states while decreasing the occurrence of the most dominant sequences in the data set also previews this decrease in the model's performance, although creating longer and more descriptive predictions.

It is also interesting to note that even when we perform the predictions using $\Lambda = 8$, the number of days throughout which the data was generated, still has a light impact on the performance values. It is observable that it fluctuates more than for the short-term predictions, however, it still maintains a reasonably similar prediction accuracy while changing this configuration.

5.5 Prototype

After the validation of the developed models performed in the previous section, we were in a position to start constructing the prototype. Given the prediction performances observed in the previous section, it is clear that the prototype would achieve a better outcome by developing it to work with sequence length $\Lambda = 4$, where the information about the last 3 locations is used to perform the prediction of only 1 future location. Besides demonstrating better prediction accuracy, the election of a sequence length of 4 cells is supported by the necessary elapsed time prior to performing predictions in a real-scenario situation. When each observation is done every 15 seconds, with $\Lambda = 4$ the model is able to start performing predictions 45 seconds after the vehicle is first detected, whereas if the prototype was to work with $\Lambda = 8$ the predictions where going to be possible only after 75 seconds. The difference may not seem that meaningful, but since we are working with a scenario where the considered area is narrow, it would heavily impact the number of predictions actually carried out.

On the other hand, the number of files used to train the prototype is not deterministic. Let us recall the existence of two different phases, the first one being the training of the model, and the second one the actual prediction. The ultimate goal of the prototype is to receive some given information about the previous 3 locations and output the most probable forthcoming one, or in other words, to make the prediction, therefore relating to the second phase. The training of the model is done before these estimations, and only the resulting trained model is used as an input for the last prototype's module, as explained in Section 5.1. With that said, it becomes trivial that we can use as many daily files of gathered data as we decide to train the model since it does not have any impact on the prediction module behaviour. The obtained predictions may differ when using different amounts of data, however, when replicating an identical prototype, one should perform offline validations similar to the one presented in Section 5.4, in an effort to understand the entails of using more or fewer data files, in terms of prediction accuracy. One should also keep in mind that using more data in the training phase will prolong the required time to obtain the fully trained model, not affecting however the time of each prediction made.

In terms of the number of used data files, we opted to use 8 files to cope better with the lack of data since using less files slightly deteriorates the model's performance, and using more than 8 files did not demonstrate any enhancements in the PP results. After deciding which sequence length to use, $\Lambda = 4$ in our case, we are in a position to train the model, and only then we are able to actually start running the prediction module itself. As stated before, this last prototype's module is a code developed in Python, that is meant to stay up and running for as long as the user wants. With that said, the code consists of an infinite loop which is constantly waiting to receive an input, performing the predicting, outputting the most probable cell, and then repeating the previous steps.

Let us address the program's expected inputs and outputs. Our program will require

the input to be sent via the console, in a string format. The three previous observations need to be separated by a comma, as c^1, c^2, c^3. The output is given via the same console, and it is simply one cell, c^4.

After receiving the input, there are three possible outcomes. The least expected one is that the prototype receives an invalid input, which will cause the program to abort. Most programs would discard an invalid input and return to the initial state, waiting for another one. However, since our program's objective is to share information with another application and not a user, we assumed that it would be very improbable to receive what we considered an invalid input, if not on purpose using it to abort the operation. Another possibility is to receive a request to predict an entity's mobility, which currently is, or was before, in an unknown location. As mentioned above, there are 21 RSUs in Aveiro however, the model is trained while considering only the RSUs that have already been reported. Therefore, an entity can be close to a deployed RSU, which is unknown to the prediction model. In this case, the prototype would output a 0. The last possible outcome is, obviously, receiving a valid input that is treated accordingly, where the predicted next cell will be outputted in the console. In Figure 5.9, we can find a few examples of the program's behaviour.

The only thing that was not presented yet is the actual implementation of the prototype, which will be done now. As explained in Section 5.1, the prediction prototype is composed of three modules, all developed in Python. The first one is the data gathering which has a pseudo-code presented in Listing 5.1. Firstly, we initialize and start a thread to increment the existing counters once every second, which are taken as a guide to know when more than 15 or 30 seconds have passed when constructing the trajectories. For this we used a thread since it runs in parallel with the rest of the code, thus not disrupting the report reception. This reception is coded afterwards by using the MQTT client class, and run indefinitely by a forever loop. Every time a report is received, the *on_message* function is called, which checks its information and appends it to the right trajectory, if that is the case. It outputs the file with the information about the RSUs that have already reported, designated as *'reported_RSUs_file.csv'*, as well as the daily files with the trajectories, that assume a name in the form of *'trajectories_file_dd_mm_yyyy.csv'*.

Listing 5.1: Code used in the Data Gathering Module.

```
1 import paho.mqtt.client as mqtt
2 from time import *
3 import threading
4 from datetime import date
5
6 MQTT_SERVER = "atcll-data.nap.av.it.pt"
7 MQTT_TOPIC = "mobility/feed"
8
9 MIN_TIME = 15
```

```
10 MAX_TIME = 30
11
12 def timer():
13     while(1):
14         increment_counters()
15         sleep(1)
16
17 timer_thread = threading.Thread(target = timer )
18 timer_thread.start()
19
20 client = mqtt.Client()
21 client.on_connect = on_connect
22 client.on_message = on_message
23 client.connect(MQTT_SERVER, 1884, 60)
24
25 client.loop_forever()
```

The second module is related to the partition of the trajectories present in the 8 files into sequences of length $\Lambda = 4$, as well as the model's construction and posterior training with the acquired sequences. In Listing 5.2, the corresponding pseudo-code is presented. We scheduled a task to be run everyday at 00:00, that starts this process. Firstly, by taking into account the present date, we import the data contained in the most recent 8 files, and for each one, the *add_trajectory()* function that will save all trajectories into the same list. Afterwards, the function *sequence_split()* is called, which is responsible to perform the partition of all the trajectories in the list into sequences of 4 cells. The obtained sequences are then used to train the model, which is exported in a file named *'LSTM_Lambda4_outputs1_days8.h5'*.

Listing 5.2: Code used in the Model Training Module.

```
1  import schedule
2  import time
3  import datetime
4  from datetime import date
5
6  def model_train():
7      print ("Initiating the model's training.\n")
8      today = date.today()
9
10     print ("Files to be used in the training are:\n")
11     i=0
12     while i < n_files :
```

```
13          yesterday = today - datetime.timedelta(days=i)
14          d1 = yesterday.strftime("%d/%m/%Y")
15          [day, month, year] = re.split('/', d1)
16          file_n="trajectories_file_"+day+"_"+month+"_"+year+".csv"
17          print(file_n)
18          add_trajectories(file_n)
19          i +=1
20
21      X, y = sequences_split()
22
23  print ("\n\n\nModel's training everyday at 00:00.\n")
24  schedule.every().day.at("00:00").do(model_train)
25  while True:
26      schedule.run_pending()
27      time.sleep(60)
```

The last module is the one responsible to interact with another application by performing predictions as requested. Furthermore, to keep the prototype in line with the most recent information, it was also necessary to automatize the import of the newly trained model everyday at 01:00. The module's pseudo-code can be found in Listing 5.3. Firstly, we initiated a thread due to the necessity to import the model at a specific hour. When this timer occurs, it calls the function *import_new_model()*, which is then responsible to verify if the current time is 01:00, and import the new model if it is. Important to note that this import is also performed once the module is started, in order to be able to make predictions as soon as possible. Afterwards, a function is called inside an infinite loop that waits for the input from the other application, and then responds with the correspondent predicted cell. If the input is invalid, it will return 1 which will cause the program to stop running.

Listing 5.3: Portion of the code used in the Model Prediction Module.

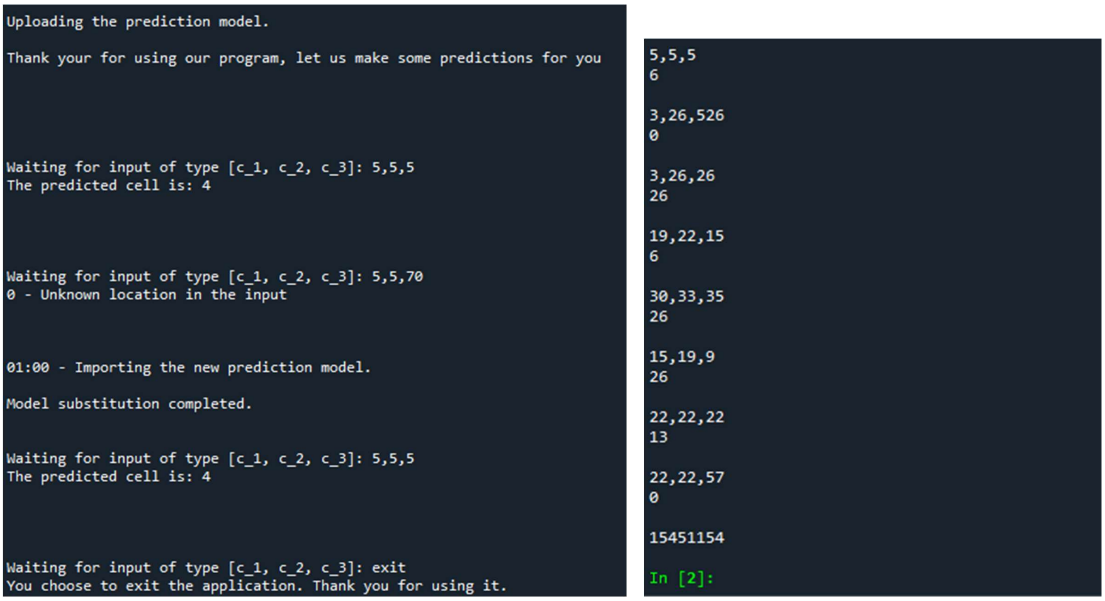
```
1  import threading
2  import datetime
3  import time
4
5  def menu():
6      #print("\n\n")
7      history = input()
8      abort = ask_prediction(history)
9
10     if abort == 1:
11         return 1
```

```
12
13 def import_new_model():
14     now = datetime.datetime.now()
15     hour = now.hour
16     minutes = now.minute
17
18     if (hour == 1) and (minutes == 0 or minutes == 00):
19         model = load_model('LSTM_Lambda4_outputs1_days8.h5')
20         with open('reported_RSUs_file.csv') as file :
21             reader = csv.reader(file)
22             reported_rsu = list(reader)
23         n_cells = len(reported_rsu)
24
25 def timer():
26     while(1):
27         import_new_model()
28         time.sleep(59)
29
30 timer_thread = threading.Thread(target = timer )
31 timer_thread.start()
32
33 while(1):
34     value = menu()
35     if(value == 1):
36         break;
```

On another note, in Figure 5.9 is possible two observe two different command line, where in Figure 5.9(a), we show how the prototype would behave if it was developed to interact with a user, and in Figure 5.9(b), the actual interaction that it has, supposedly, with third party applications. In both cases, the application was launched and used to perform some predictions, some of them using unknown locations. In both pictures the newly trained prediction model was imported, at 1 AM, despite it not being outputted on 5.9(b). To stop the program, in both cases, an invalid input was used.

5.6 Summary

We started this chapter by presenting the existing infrastructure in Aveiro and explaining the different modules that compose the prototype. This was followed by the description of the methodology employed to construct it. Both of these were explained in a generic way, as the theory behind it can be applied to the implementation of any similar application. Afterwards, we started the study on the work performed in our scenario by



(a) Simulation of the prototype's interaction with a user. (b) Real interaction with the prototype.

Figure 5.9: Interaction with the prototype through command line.

analysing the data available and resultant sequences. Later, we presented the methodology used in the model's development, which was followed by study of the prediction accuracy obtained for $\Lambda = \{4, 8\}$, on an offline manner. To finalize, we described more of the implementation of the prototype and showcased its functioning.

We come to the conclusion that our prototype is a viable solution to the issue at hand, which is also true for every other situation where this prototype is used, as long as the required adjustments are made.

6

Conclusion

We start this chapter by presenting a summary of all the work performed in the previous chapters, with its respective final remarks in Section 6.1. Afterwards, in Section 6.2 we discuss what we consider to be crucial points to be studied next to further sustain the work developed in this dissertation.

6.1 Overview

Our work aims to present the development of a mobility prediction service so that this could support other applications, with emphasis on a pre-caching service, in order to improve the management of a communications network. The proposed goal was achieved, however, its development was supported by previous studies concerning both the prediction models and the enhancements that they could bring to the network's management.

We started our work by presenting in Chapter 2 a survey of the relevant work done by other authors in previous studies. Firstly, we described why it is so important to consider the existence of mobility in the current and future communication's networks, as well as introducing some key concepts and definitions to understanding it. Then, we studied various machine and deep learning algorithms, as well as their pros and cons. Studying these algorithms helped us to choose which approaches to employ in the prediction models, allowed us to fully understand their functioning, and enlightened us on how to develop them, therefore being a crucial step. To finalize Chapter 2, we present the work carried out in multiple works where mobility has been utilized in order to enhance the management of a communications network. We presented these works in a succinct way, separated by the type of goal of each work, in order to show different studies that support the idea that mobility can be used to improve the network's performance.

In Chapter 3, our focus was to explain and evaluate the mobility prediction problem and what it entails. We started by describing, in a generic way, how we attacked the issue at hand, while introducing some key concepts such as sequences, spatial resolution, and sequence length, among others. Then, we demonstrated the methodology used to develop the required models, also in a generic manner so that in can be applied as a foundation

to any mobility prediction scenario. Afterwards, we presented the work performed for the considered scenario, which was a mobility dataset constructed from taxis travelling through Porto's city over the course of one year. Firstly, we analysed the dataset, and the sequences that were produced by separating its trajectories into fixed length sequences. To end this chapter, we analysed and compared the Prediction Performance (PP) results obtained from two different models, one based on LSTM algorithms, and the other based on a ANN approach. The comparison aimed at studying how the models' accuracy was affected by varying two factors: the sequence length and the spatial resolution.

We demonstrated that, in a general way, increasing either the sequence length or the spatial resolution will add to the prediction complexity, therefore degrading the accuracy results. We also verified that having more previous information to predict a given cell will increase the probability of correctly predicting it. Furthermore, it was shown that both proposed models were capable of achieving satisfactory prediction accuracy. Although the ANN model achieved better results in short-term predictions, the LSTM approach proved to be better on long-term predictions. On the other hand, in terms of computational time, the ANN outperformed the LSTM model.

In Chapter 4, we investigated how a mobility prediction service could improve a communications network management, which was essential to prove the applicability of our work. To further support it, our intention was to demonstrate these enhancements in two different scenarios. Therefore, we started by presenting both mobility datasets, that were completely different. The first one, for the Porto's dataset, represented more challenging predictions since the mobile entities were taxis, which move more randomly. However, this dataset was composed of an abundant amount of information which was beneficial in the model's training phase. On the other hand, the second dataset concerned buses travelling trough Aveiro, which presents a highly predictable mobility behaviour. Nonetheless, the information in this dataset is scarce, which hampers our ability to effectively train the model. The characterization of the datasets was performed after its presentation. Then we described the developed models, using both LSTM and ANN approaches, which were followed by the analysis of their prediction accuracy in order to validate their capacities. To finalize, we demonstrated the actual benefits brought by mobility prediction to a pre-caching mechanism. This was achieved by emulating the data download time when a pre-caching mechanism is implemented and when it is not. Without the pre-caching mechanism, the download is always done from the Cloud. When the pre-caching is considered, the data can be stored in advance on the RSUs to which the end-user will connect next, if the mobility prediction was performed correctly. Therefore, taking into account the prediction accuracy values previously obtained, we simulated the data's downloads from the connected RSU to the OBU when the prediction was correct, and from the Cloud to the OBU when it was not. These simulations allowed us to calculate and analyse the data retrieval time for both scenarios, if a pre-caching mechanism was implemented.

In Chapter 4, we have proven that a model could achieve a high prediction accuracy even when trained with low amounts of data, if the mobile entities' mobility is not very

random, which was not yet demonstrated in the previous chapter since that scenario did not suffer from this issue. Afterwards, we verified that the data retrieval times would decrease if a pre-caching mechanism was implemented in both communications networks, which ultimately confirms the applicability of the mobility prediction service.

In Chapter 5 we started the development of a prototype that provides the mobility prediction service. In a first moment, we presented the structure of the prototype and described the purpose of each of its modules. Then, we explained the various stages of the prototype's creation, as well as the infrastructure that is behind it. Note that this information could be applied to the development of any mobility prediction prototype which aims to interact with another application, in order to support services such as pre-caching, load balancing, and many others. One key difference from the training and prediction phases is the use of real-time data acquired through reports concerning the location of various OBUs travelling through Aveiro city. The presented model's validation is also only representative of its capabilities to perform the predictions accurately since it is done in an offline manner. Once demonstrated that the prototype was capable of receiving data dynamically, and using it to train a capable model, we describe a few examples of the code implemented in the prototype, to help the reader understand how the developed prototype was designed. To close our work, we also showcased the prototype's expected functioning when interacting with the other application.

Through the implementation of the prototype, we demonstrated that it is possible to use real-time information to build a model capable of performing the predictions when another application requests it. The prototype's implementation and evaluation was the main goal of our work.

6.2 Future Work

After the development of the prototype, as well as the studies previously performed, we can still identify some areas that could be studied more in-depth, or tasks that could be developed from scratch. Three of these are presented next.

- In Chapter 2, we surveyed various machine learning algorithms. However, throughout our work, we only studied LSTM and ANN models. With that said, it could be interesting to verify the capabilities of other machine learning approaches such as ANs in mobility prediction scenarios.

- On a different note, we demonstrated the network enhancements brought by mobility prediction in a theoretical way. It would be interesting to implement a similar scenario to the ones described in a network emulator and then observe the improvements brought to the network's management by the implementation of the mobility prediction. This would further verify the benefits anticipated in this dissertation.

- And lastly, we presented the developed prototype that aims to work in parallel with another application that implements the surveyed services. Therefore, it would be interesting to develop an application that performed one of these services, such as pre-caching, and observing the functioning of both applications while serving each other.

Bibliography

[1] N. Abani, T. Braun, and M. Gerla. "Proactive Caching with Mobility Prediction under Uncertainty in Information-Centric Networks". In: *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ICN '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 88–97. ISBN: 9781450351225. DOI: 10.1145/3125719.3125728 (cit. on pp. 28, 30).

[2] H. Abu-Ghazaleh and A. S. Alfa. "Application of mobility prediction in wireless networks using Markov renewal theory". In: *IEEE Transactions on Vehicular Technology* 59 (2 Feb. 2010), pp. 788–802. ISSN: 00189545. DOI: 10.1109/TVT.2009.2037507 (cit. on p. 29).

[3] C. Annual and I. Report. *Cisco annual internet report*. 2020 (cit. on p. 5).

[4] S. Badillo et al. "An Introduction to Machine Learning". In: *Clinical Pharmacology and Therapeutics* 107 (4 Apr. 2020), pp. 871–885. ISSN: 15326535. DOI: 10.1002/cpt.1796 (cit. on pp. 21, 22).

[5] B. Baron et al. "Mobility as an alternative communication channel: A survey". In: *IEEE Communications Surveys and Tutorials* 21 (1 Jan. 2019). ISSN: 1553877X. DOI: 10.1109/COMST.2018.2841192 (cit. on pp. 6, 8, 26).

[6] G. Bonaccorso. *Machine Learning Algorithms*. Packt Publishing Ltd., July 2017. ISBN: 978-1-78588-962-2 (cit. on pp. 9, 11).

[7] X. Cai et al. "Prediction based energy efficient mobile positioning". In: *2014 IEEE International Conference on Communications (ICC)*. 2014, pp. 2593–2598. DOI: 10.1109/ICC.2014.6883714 (cit. on pp. 26, 30).

[8] M. K. Dahouda and I. Joe. "A Deep-Learned Embedding Technique for Categorical Features Encoding". In: *IEEE Access* 9 (2021), pp. 114381–114391. DOI: 10.1109/ACCESS.2021.3104357 (cit. on p. 33).

[9] N. Devarakonda et al. "Intrusion Detection System using Bayesian Network and Hidden Markov Model". In: *Procedia Technology* 4 (2012), pp. 506–514. ISSN: 22120173. DOI: 10.1016/j.protcy.2012.05.081 (cit. on p. 14).

[10] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (cit. on p. 36).

[11] H. Farooq and A. Imran. "Spatiotemporal mobility prediction in proactive self-organizing cellular networks". In: *IEEE Communications Letters* 21 (2 Feb. 2017), pp. 370–373. ISSN: 10897798. DOI: 10.1109/LCOMM.2016.2623276 (cit. on pp. 27, 30).

[12] M. Grossglauser and D. N. Tse. "Mobility increases the capacity of ad hoc wireless networks". In: *IEEE/ACM Transactions on Networking* 10 (4 Aug. 2002), pp. 477–486. ISSN: 10636692. DOI: 10.1109/TNET.2002.801403 (cit. on p. 8).

[13] A. A. Hasbollah, S. H. S. Ariffin, and N. E. Ghazali. "Optimal Forwarding Probability for Vehicular Location Prediction Handover Algorithm". In: *Modeling, Design and Simulation of Systems*. Ed. by M. S. Mohamed Ali et al. Singapore: Springer Singapore, 2017, pp. 369–380. DOI: 10.1007/978-981-10-6502-6_33 (cit. on pp. 29, 30).

[14] L. Huang, L. Lu, and W. Hua. "A Survey on Next-Cell Prediction in Cellular Networks: Schemes and Applications". In: *IEEE Access* 8 (2020), pp. 201468–201485. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3036070 (cit. on p. 25).

[15] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks". In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*. 2008, pp. 791–798. DOI: 10.1109/COMSWA.2008.4554519 (cit. on p. 62).

[16] A. D. Ip, R. Oliveira, and L. Irio. "Vehicular Mobility Analytics". MSc Thesis. NOVA School of Science and Technology, 2021 (cit. on pp. 31, 34, 35).

[17] L. Irio and R. Oliveira. "A Comparative Evaluation of Probabilistic and Deep Learning Approaches for Vehicular Trajectory Prediction". In: *IEEE Open Journal of Vehicular Technology* 2 (Mar. 2021). DOI: 10.1109/ojvt.2021.3063125 (cit. on pp. 31, 34, 35, 50).

[18] L. Irio et al. "An Adaptive Learning-Based Approach for Vehicle Mobility Prediction". In: *IEEE Access* 9 (2021). ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3052071 (cit. on pp. 13, 14).

[19] H. Kaaniche and F. Kamoun. "Mobility Prediction in Wireless Ad Hoc Networks using Neural Networks". In: arXiv, Apr. 2010. DOI: 10.48550/ARXIV.1004.4610 (cit. on pp. 26, 30).

[20] Y. Kim et al. "Structured Attention Networks". In: arXiv, Feb. 2017. DOI: 10.48550/ARXIV.1702.00887 (cit. on p. 24).

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25 (Jan. 2012). DOI: 10.1145/3065386 (cit. on p. 20).

[22] N. P. Kuruvatti, A. Klein, and H. D. Schotten. "Prediction of Dynamic Crowd Formation in Cellular Networks for Activating Small Cells". In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1–5. ISBN: 9781479980888. DOI: 10.1109/VTCSpring.2015.7146028 (cit. on pp. 27, 30).

[23] D. A. Levine, I. F. Akyildiz, and M. Naghshineh. "A Resource Estimation and Call Admission Algorithm for Wireless Multimedia Networks Using the Shadow Cluster Concept". In: *IEEE/ACM Transactions on Networking* 5.1 (1 1997), pp. 1–12. DOI: 10.1109/90.554717 (cit. on pp. 29, 30).

[24] Y. Li. "Deep Reinforcement Learning: An Overview". In: arXiv, Oct. 2017. DOI: 10.48550/ARXIV.1701.07274 (cit. on p. 15).

[25] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. iii).

[26] J. Loy. *Neural Network Projects with Python*. Packt Publishing Ltd., Feb. 2019. ISBN: 978-1-78913-890-0. URL: www.packtpub.com (cit. on pp. 9, 16, 23).

[27] M. Mandour et al. "Handover Optimization and User Mobility Prediction in LTE Femtocells Network". In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. 2019, pp. 1–6. DOI: 10.1109/ICCE.2019.8662064 (cit. on pp. 28, 30).

[28] S. Manzoor et al. "Leveraging mobility and content caching for proactive load balancing in heterogeneous cellular networks". In: *Transactions on Emerging Telecommunications Technologies* 31 (2 Feb. 2020). ISSN: 2161-3915. DOI: 10.1002/ett.3739 (cit. on pp. 27, 30).

[29] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine Learning: An Artificial Intelligence Approach*. 1983. ISBN: 978-3-662-12407-9. DOI: 10.1007/978-3-662-12405-5 (cit. on p. 10).

[30] P. Ongsulee. "Artificial Intelligence, Machine Learning and Deep Learning." In: *2017 15th International Conference on ICT and Knowledge Engineering (ICTKE)*. 2017, pp. 1–6. DOI: 10.1109/ICTKE.2017.8259629 (cit. on p. 20).

[31] R. Pascanu, T. Mikolov, and Y. Bengio. "On the difficulty of training Recurrent Neural Networks". In: *30th International Conference on Machine Learning, ICML 2013* (Nov. 2012) (cit. on p. 22).

[32] P. R. Pereira et al. "From delay-tolerant networks to vehicular delay-tolerant networks". In: *IEEE Communications Surveys and Tutorials* 14 (4 2012), pp. 1166–1182. ISSN: 1553877X. DOI: 10.1109/SURV.2011.081611.00102 (cit. on p. 9).

[33] P. S. Prasad and P. Agrawal. "Mobility prediction for wireless network resource management". In: *2009 41st Southeastern Symposium on System Theory*. 2009, pp. 98–102. ISBN: 9781424433254. DOI: 10.1109/SSST.2009.4806853 (cit. on pp. 29, 30).

[34] C. Raffel and D. P. W. Ellis. "Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems". In: *arXiv* (Dec. 2015). DOI: 10.48550/ARXIV.1512.08756. URL: https://arxiv.org/abs/1512.08756 (cit. on p. 25).

[35] P. Rito et al. *Aveiro Tech City Living Lab: A Communication, Sensing and Computing Platform for City Environments*. 2022. DOI: 10.48550/ARXIV.2207.12200. URL: https://arxiv.org/abs/2207.12200 (cit. on p. 45).

[36] D. A. Roberts, S. Yaida, and B. Hanin. *The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks*. Cambridge University Press, 2022. DOI: 10.1017/9781009023405 (cit. on pp. 15, 20).

[37] S. Salamat et al. "Workload-Aware Opportunistic Energy Efficiency in Multi-FPGA Platforms". In: *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2019, pp. 1–8. DOI: 10.1109/ICCAD45719.2019.8942115 (cit. on p. 12).

[38] A. Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D. Nonlinear Phenomena* (Mar. 2020). ISSN: 01672789. DOI: 10.1016/j.physd.2019.132306 (cit. on p. 22).

[39] K. Xu et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: arXiv, Feb. 2015. DOI: 10.48550/ARXIV.1502.03044. URL: http://arxiv.org/abs/1502.03044 (cit. on p. 24).

[40] L. Yao et al. "A Pre-caching Mechanism of Video Stream Based on Hidden Markov Model in Vehicular Content Centric Network". In: *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*. 2020, pp. 53–58. ISBN: 978-1-7281-9216-1. DOI: 10.1109/HotICN50779.2020.9350830 (cit. on pp. 27, 28, 30).

[41] K.-L. Yap, Y.-W. Chong, and W. Liu. "Enhanced handover mechanism using mobility prediction in wireless networks". In: *PLOS ONE* 15 (1 Jan. 2020), e0227982. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0227982 (cit. on pp. 28, 30).

[42] A. Zappone, M. D. Renzo, and M. Debbah. "Wireless Networks Design in the Era of Deep Learning: Model-Based, AI-Based, or Both?" In: *IEEE Transactions on Communications* 67.10 (2019), pp. 7331–7376. DOI: 10.1109/TCOMM.2019.2924010 (cit. on pp. 11, 17).

[43] C. Zhang, P. Patras, and H. Haddadi. "Deep Learning in Mobile and Wireless Networking: A Survey". In: *IEEE Communications Surveys and Tutorials* 21 (3 2019), pp. 2224–2287. ISSN: 1553877X. DOI: 10.1109/COMST.2019.2904897 (cit. on p. 25).

[44] H. Zhang and L. Dai. "Mobility Prediction: A Survey on State-of-the-Art Schemes and Future Applications". In: *IEEE Access* 7 (2019), pp. 802–822. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2885821 (cit. on p. 13).

[45] V. Zhou. *Machine Learning for Beginners: An Introduction to Neural Networks*. July 2019. URL: https://victorzhou.com/blog/intro-to-neural-networks/ (cit. on p. 18).