UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Data analytics for mobile traffic in 5G networks using machine learning techniques

Hoang Duy TRINH

UNIVERSITAT POLITECNICA DE CATALUNYA

PH. D. THESIS

Data Analytics for Mobile Traffic in 5G Networks using Machine Learning Techniques

Author:
Hoang Duy TRINH

Advisors:
Dr. Paolo DINI
Dr. Lorenza GIUPPONI

Tutor:
Dr. Miquel SORIANO

A thesis submitted in fulfillment of the requirements for the Doctoral degree

in the

Department of Network Engineering

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Abstract

Hoang Duy TRINH

Data Analytics for Mobile Traffic in 5G Networks using Machine Learning Techniques

This thesis collects the research works I pursued as Ph.D. candidate at the Universitat Politecnica de Catalunya (UPC). Most of the work has been accomplished at the Mobile Network Department Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). The main topic of my research is the study of mobile network traffic through the analysis of operative networks dataset using machine learning techniques. Understanding first the actual network deployments is fundamental for next-generation network (5G) for improving the performance and Quality of Service (QoS) of the users. The work starts from the collection of a novel type of dataset, using an over-the-air monitoring tool, that allows to extract the control information from the radio-link channel, without harming the users' identities. The subsequent analysis comprehends a statistical characterization of the traffic and the derivation of prediction models for the network traffic. A wide group of algorithms are implemented and compared, in order to identify the highest performances. Moreover, the thesis addresses a set of applications in the context of mobile networks that are prerogatives in the future mobile networks. This includes the detection of urban anomalies, the user classification based on the demanded network services, the assistance to network optimization frameworks (e.g., mobile device wake-up schemes).

Acknowledgements

The completion of this work would not have been possible without the help of many. I would like to thank my Ph.D. advisors Dr. Paolo Dini and Dr. Lorenza Giupponi for their patience and their continuous advice during my staying at CTTC. Their precious guidance and moral support encouraged me during the hard work times. This gave me the opportunities for personal growth and for the achieved results presented in this thesis.

I would like to thank my family and my friends from all over the world for sharing the most challenging and exciting moments of the last years together. Finally, last but not least, I would like to say a special thank to Elena, for always having been next to me.

Acknowledgements to SCAVENGE This thesis is inserted in the SCAVENGE European Training Network. To tackle the large set of challenges of 5G, SCAVENGE was funded by the European Union in the context of the project Horizon 2020. This project aims to find sustainable solutions for the future mobile network technologies. 5G is intended to open many possibilities, in terms of innovations for devices, services and new applications (e.g. IoT, M2M communications, etc.). The project is structured in such a way all the major aspects of the new networks are studied. The continuous interaction between academic research and industrial partners gives a comprehensive perspective to analyse the upcoming technologies.

Contents

List of Figures

List of Tables

List of Abbreviations

IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
CDMA	Code Division Multiple Access
ML	Machine Learning
SVM	Support Vector Machine
CDR	Call Data Record
LTE	Long Term Evolution
PDCCH	Physical Downlink Control CHannel
eNodeB	Base Station
DCI	Downlink Control Information
TTI	Transmission Time Interval
MCS	Modulation and Coding Scheme
CNN	Convolutional Neural Network
SDR	Software-Defined Radio
UE	User Equipment
TBS	Transport Block Size
RNTI	Radio Network Temporary Identifier
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
NN	Neural Network

MC Memory Cell

RB Resource Block

HTTP Hypertext Transfer Protocol

GAN Generative-Adversarial Networks

SSD Solid State Storage

HMM Hidden Markov Model

RF Random Forests

AD Anomaly Detection

AE Autoencoder

CDMA Code Division Multiple Access

CDR Call Data Record

CNN Convolutional Neural Network

DBSCAN Density-based spatial clustering of applications with noise

DCI Downlink Control Information

DL Downlink

EDA Exploratory Data Analysis

eNodeB Base Station

GMM Gaussian Mixture Model

IP Internet Protocol

IoT Internet of Things

LTE Long Term Evolution

MC Memory Cell

MCS Modulation and Coding Scheme

ML Machine Learning

MLP Multilayer Perceptron

MNO Mobile Network Operator

PDCCH Physical Downlink Control CHannel

SDR Software-Defined Radio

TBS	Transport Block Size
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
NN	Neural Network
RBM	Restricted Boltzmann Machines
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RB	Resource Block
RNTI	Radio Network Temporary Identifier
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TPR	True Positive Rate
TTI	Transmission Time Interval
UDP	User Datagram Protocol
UE	User Equipment
UL	Uplink
LSTM-AE	LSTM-Autoencoder

To Elena

Chapter 1

Introduction

1.1 Context and Motivations

The penetration of Information and Communication Technologies (ICTs) in all the parts of the society is confirmed by the statistics: at the time of writing, there are an average of 6.8 billion of mobile subscribers and 750 million households connected to the Internet [1]. These unprecedented numbers of connected devices increase the mobile traffic and, at the same time, increment the complexity of the systems required to manage such amount of information. The increasing volume of exchanged data, together with new requirements for higher peak data-rates, improved reliability and reduced latency in fifth generation (5G) networks pose new challenges to telecommunications researchers and operators for increasing the system performance and, contemporaneously, improving the network power efficiency. In fact, it is expected that the 5G networks will support 1000 times more traffic, 10 times more users, and improved energy efficiency with respect to the actual 4G technology [2].

Starting from 4G, the concept of Self-organizing Networks (SONs) has been a key driver for improving Operations, Administration, and Maintenance (OAM) activities [3]. SON has been introduced by 3GPP in Release 8 [4] and aims at reducing the cost of installation and management of 4G and future technologies, bringing intelligence and autonomous adaptability into cellular networks. At the same time it points at increasing the mobile network automation and minimizing the human intervention in the cellular network management through the capability of configuring, optimizing and healing itself.

In order to enable the concept of SON, several methods like Markov models [5], fuzzy controllers [6] and genetic algorithms [7] have been applied to provide intelligence to cellular networks. One problem that arises, however, is that as the techniques get more complex, more data is required to the algorithm to meet its design requirements. Moreover, many of the solutions require expert engineers to analyze data and adjust system parameters manually in order to optimize or configure the network. Some other methods also require expert personnel on site in order to fix certain problems, when detected. All these solutions are extremely ineffective and costly to mobile operators and, although operators collect a huge amount of mobile data daily, it is not being used at its full potential. [8]

Within this challenging context, latest years have seen Machine Learning (ML) as one of the most potential and promising tool to investigate a large set of complex problems. ML techniques, in particular Deep Learning (DL) algorithms, have obtained tremendous impacts in many research fields (e.g. computer vision, speech synthesis, machine translation) [9], becoming in a relatively short amount of time the state-of-the-art framework for a wide range of problems. Self-driving car and facial recognition applications are examples of businesses that received a major impact from the development of ML technologies [10]. The rapid growth of ML is directly correlated with the technology evolution: training neural networks has become relatively fast and cheaper, thanks to more powerful hardware, like Graphical Processing Units (GPUs) and thanks to specifically designed Tensor Processing Units (TPUs) for Deep Learning [11]. Moreover, the rapid growth of the AI community has been beneficial to the development of many open-source libraries and tools (e.g. TensorFlow, Keras, PyTorch, fast.ai), that helps many researcher in the implementation and deployment of ML algorithms.

However, one of the requirements for developing valuable ML algorithms, is the existence of qualitative and measurable data: in the case of computer vision, or speech recognition, there exist a large pool of datasets that represents a shared benchmark where researchers can compare the proposed solutions. Examples are the MNIST handwriting dataset [12] or ImageNet [13], which contains more than 20,000 categories of images. In the field of wireless and mobile communications, there is not a similar counterpart. In fact, retrieving data from mobile networks it is more difficult and Mobile Network Operators (MNOs) are not always favourable to release data due to various reasons, including security and users privacy issues.

There are some exceptions, like the Data for Development (D4D) challenge from the Orange group [14], which made available anonymous data extracted from operative networks in Senegal and Ivory Coast to research laboratories. However these data are in general aggregated and do not allow deep insight into the operator's network. The lack of extensive datasets represents a major limitation for the advancement of the ML based network management research. Some databases are available ([15]), providing a precious insight in mobile network operators, but still does not give sufficient characterization on the behaviour of the network, since it does not contain detailed information on which Internet protocols and which technologies are involved in the specific communication sessions. To assist resource allocation, quality of service management and other services (e.g. anomaly detection), we need to have finer quality data, accessing directly to the network channel information exchanged between the users and the associated base stations. This allows to have access not only to aggregate base station statistics, but also to more information derived from the radio protocols, such as the resource block allocation and the link adaptation mechanism of the system, which are valuable for the above mentioned network management issues.

1.2 Objectives and Contributions

The work in this thesis is conducted in a data-driven way, and it focuses on understanding the enormous amount of information exchanged in the mobile network. Most of the data transmitted in the network can be re-utilized to discover and understand hidden structure and implicit correlation between the different parts of the system. This information can be merged with data gathered from other sources, for example using information on the typical users behaviors or on the land-use of different geographical areas. The combination of heterogeneous types of sources can be studied to enable a strong analysis and a deeper knowledge on the network and on its users.

The objective of this thesis is to characterize the users mobile traffic, the application usage and their traffic patterns. The analysis of the temporal and the behavioural statistics of the network is then needed to individuate the area for improvement in terms of efficiency and user QoS. The exploitation of a large amount of information allows to improve the performance of the network itself but also to solve a set of problems (e.g. anomaly detection, user classification) that can affect the network infrastructure.

The work starts from studying datasets that come from the actual mobile network deployments and to derive new models that may be subsequently used for the design network optimization frameworks and respond to a multitude of networking problems like resource allocation, energy saving and security, to name a few. The initial part of the work has been devoted to understand the most suitable datasets to be used in the analysis. To this end, it is important to identify different scenarios in order to cover all the possible use cases in the next generation mobile network. The data to be used can be acquired through the tools presented in the next chapters or synthetically reproduced by network simulators.

Another possibility is to find free available datasets on the Internet. The major concerns is related to the quality of the available data: most of public datasets contain irrelevant information without details on the specific technologies involved or they consist of aggregate of statistics. Thus, granularity and resolution of the dataset become discriminant characteristics for the dataset to be considered. In particular, it is possible to use a sniffer, as the one described in [16], to collect raw communication traces exchanged by the users and the associated eNodeB. This would allow to have access to millisecond statistics, that represent valuable information to describe the users' traffic at a network level, and enable the characterization of the users based on the different demanded resources.

The high level objective is to make the networks more self-aware, by exploiting the information already available in the network to gain experience in the network management, which allow to automatize the decision processes. Additionally, the use of pattern recognition and machine learning algorithms enables the design of algorithms for the prediction of the user mobility and the traffic load, in order to support network management strategies. The study of neural networks is performed to evaluate the effectiveness of these algorithms to predict the traffic patterns for a given mobile network cell. Derived models of these patterns, that are used to drive network policy algorithms, and the quantitative evaluation of network related applications, such as anomaly detection and user classification, are major contributions of this work.

The tasks that this thesis addresses can be summarized as follows:

1. **state-of-the-art study** on mobile traffic modeling, current standardization of 5G and machine learning algorithms;
2. **collection of a new dataset** using reliable tools for creating a comprehensive database on mobile network traffic to be studied;
3. **analysis of mathematical tools**: study of algorithms and ML techniques that can be exploited for the mobile traffic characterization; understanding if the learning process can be supervised or not;
4. **understanding the network traffic**: study the dataset to infer knowledge about the users and the resource scheduling mechanisms; identify opportunities for improvement and optimization;
5. **modeling and prediction of the traffic**: enabling a precise traffic prediction for aggregated traffic profiles using ML methods;
6. **anomaly detection**: identify urban anomalies in given traffic profiles related to crowded events and find efficient algorithms for recognition; extend the anomaly detection with semi-supervised procedures for unseen anomalies;
7. **classification of mobile services and apps**: extract user sessions from LTE Physical Control Channel (PDDCH) and use them to classify the user traffic based on two levels: demanded service and used applications; use the classifier to produce unsupervised traffic decomposition and profiling for given base stations;
8. **energy-efficient optimization**: based on the prediction and traffic modeling, serve the acquired knowledge to network energy optimization that would include energy harvesting sources for powering the network and proactive scheduling of the resources among connected users.

1.3 Thesis Organization

This section gives a brief outline of how the thesis is organized and which topics are presented and discussed in the next chapters. Chapters 2 and 3 are introductory and they serve as a baseline on the study of mobile networks from real traffic datasets and on the fundamentals of machine learning. The contributions are presented starting from Chapter 4, which describes in details the measurement campaign for the data collection. Chapters 5, 6, 7 and 8, include published works with results and applications derived from the collected dataset.

Chapter 2

This Chapter presents the related works on the study of the mobile network traffic using real datasets. A brief description of available datasets is given from an academic perspective. We present related works on the temporal characterization of the traffic as time-series and a literature review for anomaly detection and users' traffic classification using machine learning approaches.

Chapter 3

This Chapter provides the necessary background information regarding the machine learning algorithms implemented in this thesis. It starts describing a general taxonomy of ML based on the type of available data and labels. Then, a short introduction on Neural Networks and Deep Learning structures is given. For complete reference, the reader is suggested to refer to textbooks [17, 18].

Chapter 4

In this Chapter we explain why we need to collect a new database for studying the mobile networks. We describe the motivations and the challenge that the data collection system poses. Also, we give the details on which is our setting and tool used during the measurement campaign to capture an extensive database of mobile traffic data. This represents a major contribution of the thesis.

Chapter 5

This Chapter presents the analysis and the modeling of the traffic starting from the collected data traces. Statistical analysis and predictive modeling are given. The results reported in this Chapter are presented in the following papers:

- Trinh, H. D., Bui, N., Widmer, J., Giupponi, L., & Dini, P. (2017, October). **Analysis and modeling of mobile traffic using real traces**. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (pp. 1-6). IEEE.* [19]

- Rago, A., Piro, G., Trinh, H. D., Boggia, G., & Dini, P. (2019, June). **Unveiling Radio Resource Utilization Dynamics of Mobile Traffic through Unsupervised Learning**. In *2019 Network Traffic Measurement and Analysis Conference (TMA) (pp. 209-214). IEEE.* [20]

- Trinh, H. D., Giupponi, L., & Dini, P. (2018, September). **Mobile traffic prediction from raw data using LSTM networks**. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) (pp. 1827-1832). IEEE.* [21]

Chapter 6

In this Chapter we describe how to perform anomaly detection using mobile network data to identify potential urban anomalies caused by crowded events. The results reported in this Chapter are presented in the following papers:

- Trinh, H. D., Giupponi, L., & Dini, P. (2019, June). **Urban anomaly detection by processing mobile traffic traces with LSTM neural networks**. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON) (pp. 1-8). IEEE.* [22]

- Trinh, H. D., Zeydan, E., Giupponi, L., & Dini, P. (2019). **Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach**. *IEEE Access, 7, 152187-152201.* [23]

Chapter 7

This Chapter provides a methodology to identify with high accuracy the demanded services only by processing the information included in the unencrypted physical channel. Moreover, a novel wake-up scheme to enhance the energy-efficiency of 5G mobile devices is proposed to prolong the battery lifetime while reducing the buffering delay. The results reported in this Chapter are presented in the following papers:

- Trinh, H. D., Gambin, A. F., Giupponi, L., Rossi, M., & Dini, P. (2019). **Mobile Traffic Classification through Physical Channel Fingerprinting: a Deep Learning Approach** . *arXiv preprint arXiv:1910.11617.* [24]

- Rostami S., Trinh, H. D., Lagen S., Costa M.,Valkama M. & Dini, P. (2019). **Proactive Wake-up Scheduler based on Recurrent Neural Networks**. *arXiv preprint arXiv: 1910.11617.* [25]

- Rostami S., Trinh, H. D., Lagen S., Costa M.,Valkama M. & Dini, P. (2019). **Wake-up Scheduling for Energy-Efficient Mobile Devices**. *arXiv preprint arXiv: 1910.11617.* [26]

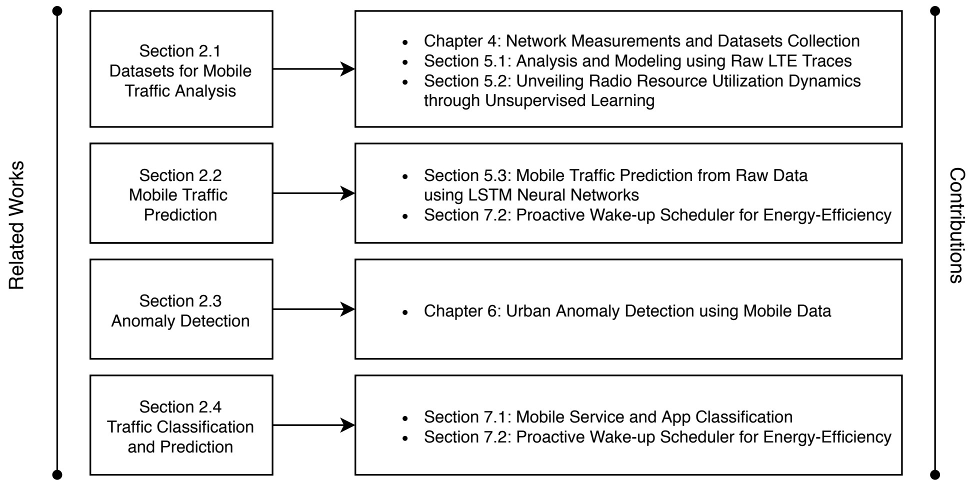
Chapter 8

This final Chapter is devoted to summarize the presented results of the thesis and to outline the future works.

Chapter 2

Related Work

This Chapter is devoted to the state-of-the-art literature on mobile traffic characterization based on data analytics and machine learning. The Chapter is organized in four sections. Each section introduces the scientific work used as background for the main contribution of this thesis explained in the next chapters, as detailed in the diagram below.



Related Works
Section 2.1 Datasets for Mobile Traffic Analysis
Chapter 4: Network Measurements and Datasets Collection
Section 5.1: Analysis and Modeling using Raw LTE Traces
Section 5.2: Unveiling Radio Resource Utilization Dynamics through Unsupervised Learning
Section 2.2 Mobile Traffic Prediction
Section 5.3: Mobile Traffic Prediction from Raw Data using LSTM Neural Networks
Section 7.2: Proactive Wake-up Scheduler for Energy-Efficiency
Section 2.3 Anomaly Detection
Chapter 6: Urban Anomaly Detection using Mobile Data
Section 2.4 Traffic Classification and Prediction
Section 7.1: Mobile Service and App Classification
Section 7.2: Proactive Wake-up Scheduler for Energy-Efficiency
Contributions

Figure 2.1: Thesis organization.

First, in Section 2.1, we give a brief description of examples of available datasets that come from operative mobile network deployments. We explain which are their main characteristics and their limitations. Then, in Section 2.2, we present related works on the temporal characterization of the traffic as time-series and models for the prediction of the future samples. Next, in Sections 2.3 and 2.4, we give a literature review on machine learning approaches for two important applications in mobile networks: anomaly detection and users' traffic classification. In particular, we present a comprehensive comparison based on different learning methods and on the type of analyzed traffic.

2.1 Datasets for Traffic Analysis in Mobile Networks

Understanding the utilization of the actual network resources is fundamental for building solid models over which working and proposing solutions to target *efficiency* in mobile networks. With the advent of new 5G paradigms and the tremendous increase of the Internet usage, there is the need for finding efficient *radio resource management* and *network planning* solutions that will further exploit and extend the actual resources, in order to provide an ubiquitous system to all the users [27]. In this context, information on the users' traffic profiles and on the network usage patterns becomes essential during the phases of planning and of deployment of the network. This translates into a more efficient allocation of the resources that can help to mitigate the effects of the increasing costs incurred by the network operators to tackle the expected upsurge of the Internet demands.

However, for research and academic communities, it is very challenging to get access to *real mobile data* extracted from the network: mobile network operators rarely release full datasets of the mobile traffic due to problems concerning, for example, the *subscribers privacy*. Users traffic data are not always made available or they can be found but with very limited information. Typically the available datasets are originally collected by network operators for billing and monitoring purposes at the cellular network operator's side.

By applying *anonymization* and *aggregation*, to preserve the customers' privacy, some datasets have been released for research purposes. One example of mobile traffic dataset is the one released in the *Telecom Big Data Challenge* [28]: this dataset is released by the Italian network operator TIM and consists of aggregated traffic derived from the *Call Detail Records* (CDRs). The objective of this context was to enable big data operations to find meaningful insights in the mobile traffic related to major Italian cities, including Milan and Rome. In the released dataset, the operator made available information on the exchanged text, voice and data, which are mixed, without additional information on the used technology. Short information is given on which type of base station the users are attached to. Also, information on the utilization of the resources at a physical level is not disclosed and therefore, no scheduling optimization of the radio resources can be assessed.

Another well-known example is the *Data for Development (D4D) challenges* [14], released by Orange, a French-based network operator who also operates in some African countries. In this case, the challenge aims at employing mobile traffic datasets towards the development of Ivory Coast and Senegal. The participants of this challenge are provided with traffic dataset that reflect users activities over the network. The mobile traffic dataset is derived from a CDR database with information about the Orange subscribers. Information related to customers' contracts at Orange during the data collection phase are omitted, and all the customers' identifiers are replaced with randomly generated numbers, to prevent a direct mapping between their identities and their mobile phone activities. Due to technical data collection problems, some information is missing from the dataset. Note that in this challenge the main focus was not primarily the study of the network related issues and optimization. In fact, most of the works using this dataset provided social insights and related solutions regarding, for example, the genesis of millet prices in Senegal [29], the ubiquitous sensing for mapping poverty in developing countries [30] and the estimation of food consumption and poverty indices with mobile phone data [31].

2.1.1 Dataset Characteristics

To be fruitful from an academic perspective, a mobile network dataset needs to comply with a set of well-defined characteristics. A comprehensive survey about mobile datasets can be found in [32]. The available mobile traffic datasets can be very different in terms of temporal granularity, aggregation level, and type of traffic. This is a direct consequence of how the data has been collected, and which is operator's goal in case the data is publicly released. As broad categorization, datasets can be classified according to the next following characteristics:

Network vs User side perspective: A first distinction is given by which kind of characterization the dataset enables to assess. Generally, we can identify two major perspectives:

1. An aggregate point of view, where the overall traffic, relative to all users accessing the network within a certain geographical area, is analyzed;
2. An individual mobile user point of view, where the behavior of each customer or mobile device is accounted for by itself.

In general, the closer the data collection is to the user, the more detailed is the information we can obtain. However, measuring the data at the users' side, generally implies significant efforts to cover a network-wide perspective. To this end, a viable solution consists of crowd-sourcing methods. For example, network information can be collected with applications installed in the user's device. These information are periodically uploaded to a server to form a continuously updated database with network-related information [15]. An example of user-side reporting the network measurements is given by OpenSignal [33].

In some cases, operators can consider network-oriented aggregation schemes, in order to avoid possible privacy problems resulting from sharing per-user information. As an example, network operators can gather all users communication data for each cell on a periodic temporal basis. Such a scheme enable networking analysis that allow to characterize usage patterns over the network [34]. From the perspective of users, aggregation schemes allow to hide users identifiers and to protect their privacy. However, when sharing per-user data, operators may provide coarse-grained information, which can limit their usefulness in the study of the network.

Temporal granularity: The temporal granularity is of extreme importance for the potential subsequent analysis of the dataset. A high-temporal resolution dataset enables a deeper study of the dataset and allows for the identification of potential optimization solutions. Starting from 4G, the communications between network and user's devices is expected to be in the order of milliseconds. Therefore, to be able to perform a complete analysis and to provide optimal solutions, the data collection should comply with the characteristics of the specific technology involved. A more coarse-grained dataset still can present a general overview of the network utilization (e.g. statistics of users' data volume per month, average speed of users per operator, etc.), but cannot be used for general academic purposes. As consequence, a higher resolution brings also more data volume and more complexity to manage.

Spatial granularity: The spatial granularity is fundamental to identify potential patterns and relationships between the users' behaviours and their usage of the network. Having specific descriptions of the areas where the network information are gathered, makes possible to formulate localized and optimized solutions. With the advent of 5G, the use-cases and the users' scenarios are multiples (e.g. high-speed scenario, rural and sub-urban areas) and optimized configurations of the network are needed to satisfy the requirements in terms of speed and network coverage. To this end, understanding for example the relationships between the user's activities, and discovering recurrent usage patterns can be fruitful for a smarter optimization of the network. However, in some cases, operators consider the position of their equipments as sensitive information. Consequently, forced by regulations, they may provide only an approximation of their locations, as done in the Orange D4D dataset, where they applied such a technique to share the positions of the base stations.

Type of Traffic: The heterogeneity of datasets is also observed in terms of traffic type. Mobile traffic operators often release CDR dataset where texts, voice and Internet data communications are measured for bill charging purposes. However, from an academic perspective, it would be important to define also the specific use-case: understanding which protocols are used (e.g. TCP or UDP), or which technology has been involved (e.g. Voice over IP (VoIP), 3G, 4G, LTE), it is of more interest from an engineering and research point of view, and then can help, for example, to identify different user classes, with different Quality of Service (Qos) requirements.

2.2 Mobile Traffic as Time Series

In literature, the mobile traffic dynamics is often modeled as a time-series representing a set of traffic features over specific time interval [35, 36, 36, 34, 37, 38]. From an analysis point of view, this representation can be used to track the temporal evolution of both aggregate and individual user traffic. Additionally, it enables different kinds of *aggregation schemes* and therefore, diverse *levels of granularity*, depending on the objective of the analysis.

Studying the spatio-temporal dynamics of users' traffic allows to understand the evolution of the aggregate demands over the access network and to find repetitive patterns. For example, it is common to identify a cyclic daily behavior characterized by a low traffic at night hours and a high traffic during day hours, across different datasets. Accordingly, mobile traffic is observed to follow a regular behavior. For example, authors in [39], while analyzing the packet data call activity over multiple cells for one week duration , notice that the traffic presents a daily repetitive behavior over weekdays, characterized by a low activity during the night and a high activity during the day. The same behavior is also perceived at different time scales in other several studies [40, 35, 41].

Although the most significant load difference is between night and day, other studies have pointed to traffic variations over different temporal scales, suggesting the need for further investigations to understand the correlation with the human activity dynamics over smaller time intervals. In [42], the authors detect variations at an hourly basis, of both calls and texts activities in two different scenarios: San Francisco, and an unnamed Chinese city. Interestingly, they identify the occurrence of two daily peaks over the Chinese dataset for calls and one peak for texts, while in the case of San Francisco, they detect only one peak for calls and no peak at all for texts. The study in [39] exploits temporal variations at an even finer granularity, of 10 minutes, in packet data call traffic. Authors detect the presence of several peaks, with the largest ones appearing mostly in the late afternoon. Significant differences are also observed for various traffic types between weekends and weekdays. In [39], notice that packet data call traffic is higher on weekdays with respect to weekends. Finally, authors in [43] capture seasonal variations in users traffic: they detect an increase of 20% in monthly data usages towards the end of the year with respect to data usages in the summer.

2.2.1 Prediction and Modeling of Mobile Traffic

Predictive analysis on mobile network traffic is becoming of fundamental importance for the next generation cellular network, and the *proactive* knowledge about which are the user demands, allows the system for an optimal network resource allocation. To meet the strict requirements of 5G, and its new set of applications, it is fundamental that the network becomes aware of the traffic demands and be able to predict with accuracy the user's requests .

Due to the characteristics of the available datasets, the prediction of mobile traffic patterns has been usually studied through time-series analysis methods. As consequence, most of the works treat the mobile traffic analysis using techniques such as Auto Regressive Integrated Moving Average (ARIMA) and Support Vector Regression (SVR), which are two most widely used methods to deal with time-series. Also different versions of the ARIMA model (e.g. SARIMA, ARIMAX, mixed ARIMA, Fractional-ARIMA), have been applied to wireless networks to capture the trends of the temporal evolution of the mobile traffic, [44], [41]. In particular, SARIMA (Seasonal ARIMA) is a technique to study the seasonal and cyclic patterns that are present in the traffic, in order to find the recurrent behaviour of the user's mobile demands.

However, one of the known limitations of such techniques is the poor robustness to the rapid fluctuations of the time-series, since the prediction tends to over reproduce the average of the past observed values [45]. In some cases, the data resolution can be in the order of milliseconds, allowing for a deep analysis in terms of temporal granularity: however, this implies also a rapid change in the time-series that cannot be captured by traditional methods. SVR models are also limited for the reason that the users need to determine some key parameters for the model, and there is no a structured way available for determining best values for these parameters [46]. Additionally, these methods work well with homogeneous time-series, where the input and the prediction are within the same set of values. When the dataset includes a higher number of metrics, the data structure becomes more complex and the difficulty of extracting meaningful patterns increases as well.

For these reasons, accompanied by rapid technology evolution, in recent years *Artificial Neural Networks (ANNs)*, and in particular *Recurrent Neural Network (RNNs)*, have shown outstanding results for time domain problems and sequential data. They have been heavily adopted for text prediction and machine translation [47], becoming the state of the art approach for this type of problems. These neural network structures can fit the problem of mobile traffic prediction, in particular, when data consists of multivariate features and presents heterogeneous, non-linear information about the users communications. Moreover, the time-domain characteristic of the mobile traffic can be assisted by the RNN properties, which are able to capture the temporal trends of the data.

Results in literature show that RNN networks outperform other machine learning approaches for time-series analysis in the traffic prediction. As example, *Long Short-Term Memory (LSTM)* structures have been proposed in [41] for traffic prediction. Long Short-Term Memory networks represent the most used implementation of RNN due to their complex structure, which allows them to solve the vanishing-gradient problem and learn longer term dependencies from the input data. In [41], LSTM has been used to study a dataset that consists of the spatio-temporal distribution of the mobile traffic from different base stations. Spatial correlation has been used to highlight similarities between neighbouring base stations. Even though LSTM has been applied for the traffic prediction, in this case the input data consider only one feature (e.g. the spatially distributed traffic), and only a one-step prediction. A multi-step prediction would be able to predict the traffic for multiple steps, providing a longer horizon time to perform potential network optimization. Moreover, considering more than one feature enables a multivariate characterization of the traffic, that would include, at the same time, the radio-resource allocation and data-rate of the communication between the device and the eNodeB.

2.3 Anomaly Detection using Mobile Network Data

With the rising capacity demand in mobile networks, the infrastructure is becoming larger, denser and more complex to manage. Therefore, a key characteristic of the next generation systems is the ability to be dynamic, programmable and reconfigurable, driven by analytics and intelligence. This feature enables automatic and autonomous operations that may address a multitude of issues, e.g. capacity planning, QoS/QoE management, outage detection and relief, energy saving, to name a few, so as to optimize network functioning, simplify its maintainance and save costs. In this context, the automatic detection of urban anomalies, like unexpected crowd gathering, is of upmost importance for government and public administration [48].

Network traffic anomalies may be defined as unusual and significant changes in the traffic of a network, which a mobile network operator (MNO) needs to timely detect to take the proper actions and maintain the right operation of its network ([49], [50]). Detecting anomaly using mobile network data represents a perfect use-case for exploiting the pervasiveness of mobile networks in urban areas. Examples of anomalies include both legitimate activities such as transient changes in the customer demand, flash crowds, and illegitimate activities such as Distributed Denial of Service (DDoS) attacks, device eavesdropping, etc. [51].

	Different Approaches		**Proposed Approach**	
	Characteristics	**Limitations**	**Advantages**	**Differences**
Traditional	Density and distance based [52, 53]	High data dependence	Use of multivariate data to increase hit rate (recall)	Higher order distinction between AD and normal traffic in crowd events
	Parametric-based [54, 55, 56]	Model dependence, difficulty dealing with high dimensional data		Contextual Anomaly rather than only point AD
Shallow Learning	Based on dataset labels: supervised [57], unsupervised [58], semi-supervised [59]	Severe class unbalance Model drift	Ability to interact with raw complex network data	Extension for high dimensional mobile network data
		Uncertainty around data model Scaling issues for large data	High Learning Capacity	Early detection capabilities
Deep Learning	Using CNN, RNN, LSTM [60] to identify anomalous behaviour	Depends mostly on training over single deep model Limited application scope (e.g. AE used most for intrusion detection)	Apply a decision function on reconstruction (or prediction) error to take into account traffic variations	Detection of contextual anomalies using two step approach and with hybrid models (e.g. LSTM-AE)
Network Datasets	CDR-driven [61, 62] Network flow (TCP connection) driven [63, 64] Social network data driven [65] Log analysis to detect suspicious behaviour [66]	Mostly proprietary data Low resolution and aggregated data Difficult to correlate with event No operational datasets	Fine-grained measurements (every 1ms TTI) Detect all the users connected to LTE eNodeB Availability of PHY Network informations (i.e. Resource Blocks, MCS index)	Based on unencrypted LTE PDCCH measurements Spatial and time context that allows for social event correlation Can be performed without MNO intervention

Table 2.1: Overview of AD techniques, datasets and comparison with the proposed approach.

2.3.1 Traditional AD Approaches

Anomaly Detection (AD) and outlier detection have been widely investigated in different research areas and there are many works that cover these issues from statistical perspectives [52, 53, 67, 68, 69, 54, 70, 63]. One of the major challenges that is encountered in all of these works is the design of a suitable model that can accurately separate normal data from unusual data points. A classification of traditional AD algorithms can be done based on four main categories: density-based, distance-based, parametric and statistical-based algorithm. In density-based and distance-based AD, spatial proximity of data points are used. Density-based spatial clustering of applications with noise (DBSCAN) [52], Isolation Forest and Local Outlier Factor (LOF) [53] are some examples of density-based AD algorithms. Distance based AD algorithms comprises adaptations of clustering algorithms, including K-NN [67, 68], and K-Means [69]. Gaussian Mixture Model (GMM) [54], Single Class SVMs [55] and Extreme value theory [56] are notable examples of parametric based approaches. Statistical tests such as *z-score* and variations [70] are examples of statistical based approaches. However, traditional approaches are not able to handle the complex nature of raw mobile control channel data.

Compared to traditional approaches, in our work we focus on AD using multivariate data analysis to increase the overall hit-rate of anomaly detection to differentiate between anomalous and normal events. Moreover, our analysis is based on contextual AD events rather than point anomalies as is usually studied with traditional AD approaches.

2.3.2 Shallow Learning based AD Approaches

Shallow learning algorithms for AD depend on the availability of the dataset and on how the algorithms are trained (using a labeled dataset, using an unlabeled dataset or using partly the dataset with samples of the majority class). In case of a labeled dataset, the algorithm can be trained supervisedly: in this case, the AD is performed as a classification task to differentiate the normal class from the anomalous class. However, there are two major issues that arise in supervised anomaly detection: first, the anomalous instances are far fewer compared to the normal instances in the training data, creating imbalanced class distributions [71, 72]. Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging.

The performance of supervised anomaly detection are in general superior to unsupervised approaches [73, 58]. However, obtaining a labeled dataset for AD is an extremely expensive operation due to the few anomalous occurrences. In a complex scenario like operative mobile networks, it is unrealistic and infeasible to set the AD problem as a supervised task. Moreover, as the dimensionality of mobile data increases, shallow learning approaches have difficulty in terms of scaling. For these reasons, unsupervised or semi-supervised approaches are more favorable. A semi-supervised statistical based method using the Call Data Records (CDRs) provided by a network operator is given in [59].

2.3.3 Deep Learning based AD Approaches

Emerging deep learning techniques recently have been applied to AD problems. A survey of such techniques within diverse research areas and application domains is provided in [60]. Application of traditional approaches in combination with deep learning have also been studied. For example, the authors in [74] have demonstrated that before training a RNN model, an initial clustering with K-NN can enhance the detection of outliers in social media. A more recent survey paper that brings together previous approaches on deep learning in the domain of wireless and mobile networking domain is given in [75]. Different deep learning architectures such as Restricted Boltzmann Machines (RBM), Autoencoders (AEs), CNN and RNN have also been considered to extract the representative features from the datasets and obtain a characterization of network traffic behaviours. For example, CNN and RNN are proven to be able to study multi-dimensional input correlation, and they are applied to model spatial and temporal characteristics of the mobile data in [76, 77]. In these cases, anomalies can be detected whenever a sample displays characteristics that are significantly isolated in the feature space. Authors in [78] combine CNN and RNN for automatic feature extraction and AD from web traffic over a one-dimensional time-series signal. Another deep learning based approach with autoencoders is used to reconstruct the input samples and it has been employed for intrusion detection in [79]. A supervised AD algorithm using variational AEs is provided in [80], in order to solve both seen and unseen anomalies. Using AE, the reconstruction error can be evaluated and those samples that show abnormal values are likely to be considered anomalies.

In some cases, the learning is done in a supervised manner using a labeled dataset, where the algorithms are trained using both classes (normal and anomalies) of traffic instances. In contrast, in this work, due to the nature of the dataset, we choose a semi-supervised deep learning approach, where only one class of samples (normal traffic) is needed to train the AD algorithms. Our proposed design comprises a stacked architecture combined with Long Short-Term Memory (LSTM) cells, which are an improved structures for RNN implementation, that are able to extract the relevant features from the multivariate input dataset collected from mobile network. In this work, we use a two-steps approach to detect anomalies. First we exploit generating hybrid models (e.g. with autoencoders). After building the deep learning model, instead of using a static threshold on the reconstruction (or prediction) error to detect the anomalies, we use first and second order statistics to calculate a dynamic error threshold that take account of the traffic variation during different hours of the day using a moving average with discrete linear convolution method.

2.3.4 Datasets used for AD Approaches

Datasets used for AD approaches are diverse and involve different data domains such as cyber-intrusion detection systems, image and speech recognition, video surveillance, industrial Internet of Things (IoT) applications, data center log analysis and analysis of cellular network and communication data. For example, a survey for anomaly detection using social network AD is given in [65]. AD methods on data center log dataset has been done to detect suspicious activities in [66].

In the area of communication mobile networks, AD has been investigated in the literature mainly based on usage of different datasets [63, 64, 62, 81, 61]. The authors in [63] follow a statistical based AD approach where the proposed change-detection algorithm is used to characterize the large scale dataset obtained from a real operational mobile network. The paper in [64] proposes a deep learning based network AD system against malicious malware forming botnets in 5G networks using network flow data features. The authors in [62] analyze CDRs of the Mobile Network Operator (MNO) to extract anomalies and predict the future traffic over anomaly-free data using neural networks. In [81], the authors adapt wavelet transformation techniques to identify cellular network anomalies related to social events. Hussain et al. in [82], investigate a semi-supervised statistical based AD approach using CDR dataset provided by the operator Telecom Italia after the Big Data Challenge 2014 competition. CDRs are also used in [61], in which Karatepe et al. analyze the dataset to find anomalies in generated CDRs using a rule based approach

Differently than previous approaches, in this thesis we use Long Term Evolution (LTE) control channel data source to identify potential anomalous events which can give fine-grained measurement level. AD on this level of measurements can also yield early detection possibilities compared to more coarse-grained measurements such as CDRs or social media analysis outcomes. Moreover, we consider combination of multiple fingerprints in the raw dataset to help us detect not just point anomalies but also contextual anomalies. This has the potential to increase the precision and hit-rate with respect to other data sources.

2.4 Mobile Traffic Classification

The categorization of network traffic into appropriate classes has many relevant uses spanning from Quality of service (QoS)/Quality of Experience (QoE) control and management, to pricing, network resource management, malware detection, and intrusion detection, to name a few. The key challenge of such classification algorithms consists in the identification, and in the subsequent computation, of a number of *representative features*. These features are then used to train algorithms that classify the data flows at runtime. Most of the surveyed approaches leverage some *domain knowledge*, which is utilized to *manually* obtain the feature set, i.e., using prior information by a human expert.

2.4.1 Learning Methods for Traffic Classification

The use of deep learning techniques has recently paved the way to automatic feature discovery and extraction, often leading to superior performance. For example, in [83] encrypted traffic is categorized through deep learning architectures, proving their better performance with respect to shallow neural network classifiers. The authors of [84] present a mobile traffic super-resolution technique to infer narrowly localized traffic consumption from coarse measurements. In detail, a deep-learning architecture combining Zipper Network (ZipNet) and Generative Adversarial neural Network (GAN) models is put forward to accurately reconstruct spatio-temporal traffic dynamics from measurements taken at low resolution. Another example is found in [85], where identification of mobile apps is performed by automatically extracting features from labeled packets through CNNs, which are trained using raw Hypertext Transfer Protocol (HTTP) requests, achieving a high classification accuracy. The work in [83, 84, 85], as the majority of the other techniques , use statistical features obtained from application or Internet Protocol (IP) level information for both service and app identification, along with UDP/TCP port numbers.

The most common classification methods in the literature leverage UDP/TCP port analysis and/or packet inspection. UDP/TCP port analysis relies on the fact that most Internet applications use well-known Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port numbers. For instance, the authors of [86] define a mobile traffic classifier as a collection of rules, including destination IP addresses and port numbers. Based on these rules, application-level mobile traffic identification is performed deploying a dedicated classification architecture within the network, and measurement agents at the mobile devices. However, port-based schemes hardly work in the presence of applications using dynamic port numbers [87].

A scheme based on deep packet inspection is presented in [88]. The authors of this paper devise a technique for Code Division Multiple Access (CDMA) traffic classification, using correlation-based feature selection along with a decision tree classifier trained on a labeled dataset (which is labeled via deep packet inspection). The algorithm in [89] extracts application layer payload patterns, and performs maximum entropy-based IP-traffic classification exploiting different Machine Learning (ML) algorithms such as Naive Bayes, Support Vector Machines (SVMs) and partial decision trees. Remarkably, payload-based methods are limited by a significant complexity and computation load [87]. Furthermore, many mobile applications adopt encrypted data transmission due to security and privacy concerns, which renders packet inspection approaches ineffective.

Classification schemes for encrypted flows utilize traffic statistics, extracting meaningful features from the observed traffic patterns. For example, the authors of [90] propose a classification system for mobile apps based on a Cascade Forest algorithm exploiting features related to: i) the mobile traces, ii) the Cascade Forest algorithm, iii) connection-oriented protocols and connection-less protocols, and iv) encrypted and unencrypted flows. Along the same lines, a classification approach that combines state-of-art classifiers for encrypted traffic analysis is put forward in [83]. Another interesting work is presented in [87], where the authors classify service usage from mobile messaging apps by jointly modeling user behavioral patterns, network traffic characteristics, and temporal dependencies. The framework is built upon four main blocks: traffic segmentation, traffic feature extraction, service usage prediction, and outlier detection. When traffic flows are short and the defined features do not suffice to fully describe the traffic pattern, Hidden Markov Models (HMMs) are exploited to capture temporal dependencies, to enhance the classification accuracy.

The authors of [91] show that a passive eavesdropper is capable of identifying fine grained user activities for Android and iOS mobile apps, by solely inspecting IP headers. Their technique is based on the intuition that the highly specific implementation of an app may leave a fingerprint on the generated traffic in terms of, e.g., transfer rates, packet exchanges, and data movement. For each activity type, a behavioral model is built, then K-means and SVM are respectively used to reveal which model is the most appropriate, and to classify the mobile apps.

Automatic fingerprinting and real-time identification of Android apps from their encrypted network traffic is presented in [92]. IP-based feature extraction and supervised learning algorithms are the basis of a framework featuring six classifiers, obtained as variations of SVMs and Random Forestss (RFs). RFs have also been considered in [93], where the authors claim that the sole use of packet-based features does not suffice to classify the traffic generated by mobile apps. As a solution, they use a combination of packet size distributions and communication patterns. Recent works exploit Neural Networks (NNs) [85, 84, 83]. In [85], mobile apps are identified by automatically extracting features from labeled packets through CNNs, which are trained using raw HTTP requests. In [83], encrypted traffic is classified using deep learning architectures (feed forward, convolutional and recurrent neural networks) for Android and iOS mobile apps, with and without using TCP/UDP ports. The authors of [84] combine Zipper Networks (ZipNet) and Generative-Adversarial Networks (GAN) to infer narrowly localized and fine grained traffic generation from coarse measurements.

A systematic framework is devised in [94] for comparison among different techniques where deep learning is proposed as the most viable strategy. The performance of the deep learning classifiers is critically investigated based on three mobile datasets of real human users’ activity, highlighting the related drawbacks, design guidelines, and challenges. Several survey papers dealing with deep learning techniques applied to traffic classification can be found in [95], [96] and [75]. The authors in [95] overview general guidelines for classification tasks, present some deep learning techniques and how they have been applied for traffic classification, and finally, open problems and future directions are addressed. The survey in [96] presents a deep learning-based framework for mobile encrypted traffic classification, reviewing existing work according to dataset selection, model input design, and model architecture, and highlighting open issues and challenges. Finally, a comprehensive and thorough study of the crossovers between deep learning and mobile networking research is provided in [75] where the authors discuss how to tailor deep learning to mobile environments. Current challenges and open future research directions are also discussed.

We stress that that most of the works in the literature, with the exception of [85, 84, 83] and [94], classify mobile traffic based on manual feature extraction and all the papers that we surveyed process network or application level data. Our work departs from prior art as we leverage the feature extraction capabilities of deep neural networks and classify mobile data gathered from the physical channel of a mobile operative network, at runtime, and without access to application data and TCP/UDP port numbers.

Chapter 3

Machine Learning Background

In this Chapter, we discuss about the Machine Learning (ML) framework applied to mobile traffic analysis. This part presents an overview of the algorithms that have been used to solve the different problems of mobile traffic characterization proposed in this thesis. The objective of this chapter is to give a taxonomy of ML algorithms and the different architectures we adopt in our proposed solutions.

First, in Section 3.1, we give a broad categorization of the ML algorithms based on the learning procedure. Then, in Section 3.2, we introduce Neural Networks-based algorithms and we present several deep learning architectures. Finally, in 3.5, we give the details on the code implementation and we list the libraries used for the software development.

3.1 ML Taxonomy

Based on the learning process, the set of ML algorithms can be broadly partitioned in supervised, unsupervised, and reinforcement learning algorithms. The application of each type of algorithm depends on the availability of labelled data and on the problem goals. Generally, when we approach a problem using ML, the first issue to be addressed is to understand which type of data we are able to provide to the algorithm and the amount of information we can use to teach the algorithm to solve the problem.

In the context of mobile networks, we observe that different problems can be addressed with different type of ML algorithm. The heterogeneity and the complexity of mobile networks make the choice broad: tons of data are generated and transmitted at different level of the communication stack, from the physical to the application layer. The choice of which information to be used is fundamental and must consider a multitude of factors, including the easiness to retrieve a large pool of data and the granularity of information.

3.1.1 Supervised Learning

In Supervised Learning, the algorithms learn using labelled data. In this case, the objective is to build a statistical model for predicting, or estimating, an output based on one or more inputs. Generally, supervised learning involves observing several examples of a random vector x and an associated value or vector y, and learning to predict y from x, usually by estimating $p(y|x)$. The term *supervised* learning originates from the view of the target y being provided by an instructor who shows the machine learning system what to do. For example, based on the collection of mobile users data sessions, we want to implement a real-time classifier that is able to identify to which category the user belongs to. If we succeed in collecting a sufficient number of samples, we can train a supervised way a ML algorithm where each session is labelled according to the specific class.

The efforts to obtain a labelled dataset must be measured with respect to the final objective: having a labelled dataset enables the implementation of supervised learning algorithms, which are more goal-defined and more precised to solve specific problems. However, in many cases this is not possible due to the complexity of the labeling operation, which may require time and manual intervention. Some examples of supervised learning tasks are the classification of users based on their activity, as done in [24], or prediction of the LTE mobile traffic [21]: in the latter case, the problem can be state as time-series prediction problem. In literature, this a well-know problem which can be resolved with classic methods like ARIMA, Seasonal-ARIMA [97], but the multidimensionality of the inputs and the complexity of the data, justify the use of ML algorithms that have shown great performance in sequential data problems like machine translation and natural language processing (NLP).

3.1.2 Unsupervised Learning

With unsupervised statistical learning, the goal is to learn relationships and structure from the input data. The algorithm receives unlabelled input with the objective to find a pattern and learn relationships from it. In this case,we let the algorithm learn by itself, without providing the correct answer to the problem we want to solve. Conversely to supervised learning, unsupervised algorithms involve observing several examples of a random vector x, and attempting to implicitly or explicitly learn the probability distribution $p(x)$, or some interesting properties of that distribution. In unsupervised learning, there is no instructor and the algorithm must learn to make sense of the data without any guide.

One of the most common unsupervised task is clustering, where the observed samples are grouped into *clusters*, according to their intrinsic characteristics. Traditional algorithms for clustering can be categorized in distance-based algorithms or density-based algorithms. In distance-based algorithms, the observed elements are grouped based on a measure of distance or similarity (usually pair-wise); in density-based approaches, instead, the clusters are formed based on the density and on the sparsity of elements in the features area (e.g. DBSCAN). With reference to the previous example, we may have access to many mobile user sessions but we may not have a defined label for each session. Instead, we are interested into grouping the users based on the characteristics of the sessions. Each session may be defined by a multitude of parameters: the increasing dimensionality can increase the complexity of the problem, making necessary to use more complex algorithms to extract the hidden features needed for the classification.

3.1.3 Semi-Supervised Learning

In Semi-Supervised Learning, the learning process is a blend between supervised and unsupervised. There is not a strict definition for this type of algorithms, but generally, the learning includes both supervised and unsupervised tasks: for example, within this category, we can put those algorithms using datasets that are only partially labeled. Typically, in practical scenarios, only a small amount of samples is labelled, and the objective is to assign a label to the unlabelled portion of data, which is normally larger.

In this work, we use a semi-supervised approach for anomaly detection on LTE traces: we want to detect the traffic associated to anomalous events. To this end, we perform supervisedly the separation between those traces that we consider not anomalous with high degree of certainty and then, we perform the features extraction in a unsupervised way using Autoencoder structures.

3.1.4 Reinforcement Learning

Reinforcement Learning (RL) is defined as a set of algorithms to solve sequential decision processes where one agent directly interact with the environment, which returns a reward through which the agent learns the optimal policy. The definition of a RL problem involves an environment that can be defined by a set of actions $\mathcal{A}$, a set of states $\mathcal{S}$, and a reward function R. The formulation of the problem is derived from Markov-Decision Processes (MDP), and mathematically the solutions are equivalent for both formulations. In the context of mobile networks, we can model a problem with Reinforcement Learning, when the network agents take actions to manage the network procedures, protocols or functions.

In this thesis, we focus on the mobile network traffic modeling and characterization rather than optimizing network precedures. Therefore, RL has not been used in the solutions presented in this work. Nevertheless, the traffic models and the prediction of the traffic given in this thesis, allows for a characterization and a simplification of the traffic states that can be exploited in related works, where RL algorithms have been implied [98].

3.2 Artificial Neural Networks

Artificial Neural Networks (ANN) aim at reproducing the complex behaviour of the human brain learning starting from a single cell unit called neuron. The popularity of ANNs is due to the fact that most of the algorithm optimization is based on the *gradient-descent* method, that can be implemented efficiently on hardware using computational graphs. To this end, specific hardware has been built to facilitate and to improve the performance of Neural Networks: GPUs and TPUs (Tensor Processing Unit) are widely available on the market to perform such operations and can be easily associated with cloud infrastructures that allow to manage large data lake and Big Data operations.

Similarly to the brain cells, the basic element of ANNs is also called neuron: in practice, the simplest neuron is represented by the Perceptron, which performs basic mathematical operations. Perceptrons can be combined to form a layer, and layers can be connected to build complex multilayers networks that can be represented as directed graph, where the nodes are the neurons and the edges are connections.

Next, we present the ANN structures that have been adopted in this work, and the analysis is limited to the description of the main architectures, without reporting the full mathematical notations. For further reading, the author suggests [17, 18].

3.2.1 Multilayer Perceptron

A single perceptron is the basic component in an ANN and it is graphically represented by a circle (Fig.3.1). Suppose our dataset consists of m input-output pairs $(x^{(i)}, y^{(i)}), i = 1, \ldots, m$: the simplest perceptron takes an input $x^{(i)} = [x_1, x_2, \ldots, x_n]$, and produces an output $\hat{y^{(i)}}(w, x^{(i)})$, performing a linear combination with its weight w_i and non-linear activation function $\sigma(\cdot)$:

$$\hat{y}^{(i)} = \sigma\left(\sum_i^n x_i w_i + b\right) \tag{3.1}$$

Generally, the single neuron operations are mathematically expressed as matrix multiplication. For this reason, it is common to add $x_0 = 1$ as part of the input vector $x^{(i)} = [1, x_1, x_2, \ldots, x_n]$ and include the bias term b in the vector of weights $w = [b, w_1, w_2, \ldots, w_n]$, so that the notation simplifies as follow:

$$\hat{y}^{(i)} = \sigma\left(w^T x^{(i)}\right) \tag{3.2}$$

Multiple perceptrons can be combined together to form a layered architecture, usually named *multi-layer perceptron*. In the simplest case, each neuron performs the same mathematical operations, using the same non-linear activation function. In many text books, the input and the output are considered, respectively, the first and

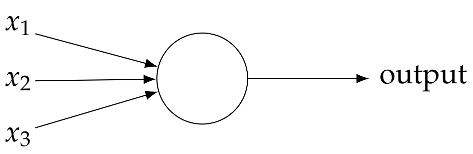


Figure 3.1: Perceptron graphical representation.

the last layers of the neural network. A graphical representation of a MLP is given in Fig.3.2. The intermediate layers are called *hidden layers*, the neurons of this layer are called *hidden neurons*, and their number defines the size of the layer. The term hidden refers to the fact that we do not have the ground truth/training value for the hidden units, in contrast to the input and output layers, both of which we know the ground truth values from $x^{(i)}, y^{(i)}$.

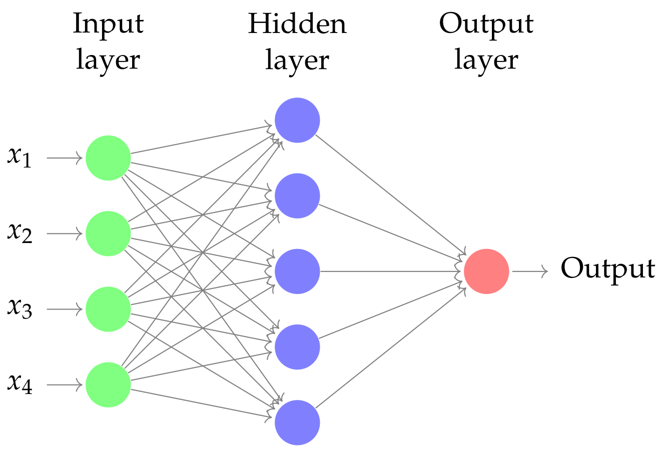


Figure 3.2: Example of fully-connected MLP architecture with 1 hidden layer and 5 hidden neurons.

Without considering the non-linear activation function, in our problem, we should have that the classes must be linearly separable by a the decision surface consisting of a hyperplane, as shown in Figure 3.3.

The task of the non-linear activation function is therefore, to extend the ability of a neural network to non-linearly separable cases. The most common used non-linear activation function is the sigmoid function:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{3.3}$$

In our case, let $z = w^T x$ be the linear combinations of input and weights. Eq. 3.3 then becomes

$$\sigma(z) = \frac{1}{1+exp(-\sum w_i x_i + b)} \tag{3.4}$$

The shape of the sigmoid function resembles the step function. One of most interesting properties of using the sigmoid as non-linear activation function is its derivative,

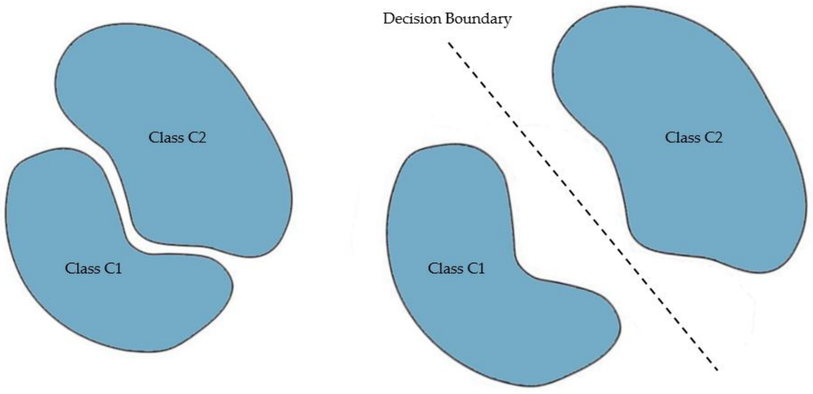


Figure 3.3: A pair of linearly separable vs. a pair of non-linearly separable patterns.

which can be written as:

$$\sigma'(z) = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma(z)(1-\sigma(z)) \tag{3.5}$$

The simple calculation of its derivative make the sigmoid function very fast to be implemented in the back-propagation algorithm, which is used to train the network.

3.2.2 Feed-Forward Networks

We refer to the ANN represented in Fig. 3.4. The ANN is also called Feed-Forward Networks since the input data is "fed" to the network (forward-pass). The operations of each layer can be denoted with a matrix of weight $W^{[l]}$, where l refers to the layer number (we consider $l = 0$ for the input layer). The dimensions of this matrix are equal to the number of inputs multiplied by the number of hidden neurons. At each layer l, we denote the linear combination with $z^{[l]}$, and the activation with $a^{[l]}$:

$$z^{[l]} = W^{[l]}x^{(i)} + b^{[l]} \tag{3.6a}$$
$$a^{[l]} = \sigma(z^{[l]}) \tag{3.6b}$$

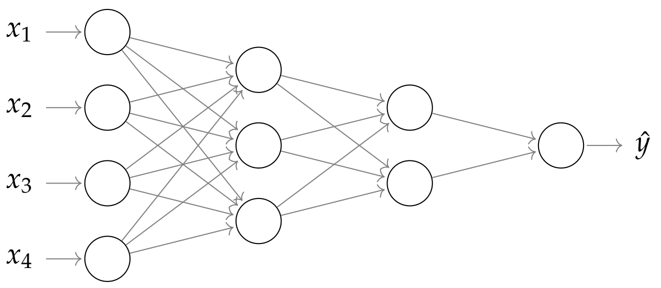
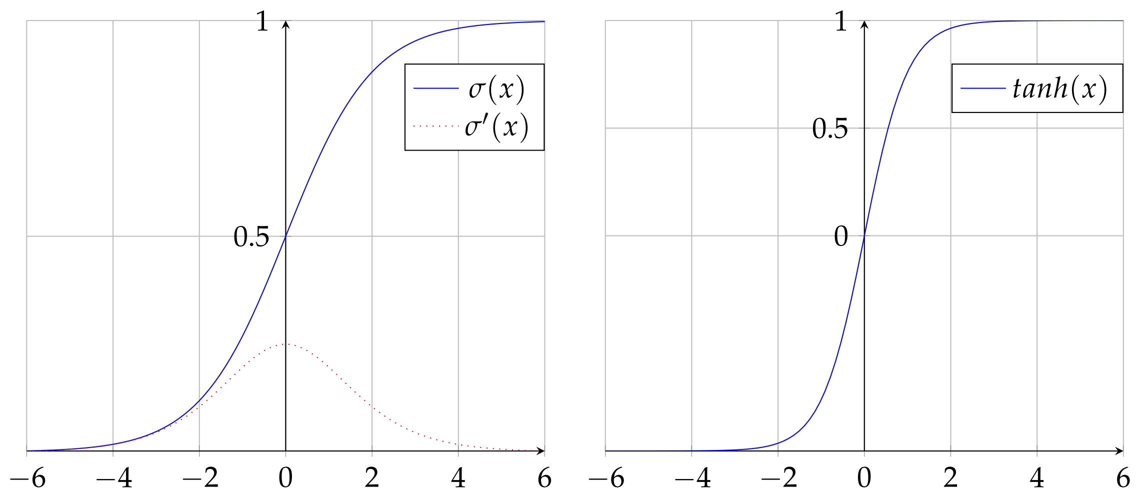


Figure 3.4: Example of ANN with 2 hidden layers with size 3 and 2.

The forward-pass consists of feeding the data from the input to the output layers (from left to right). In the example, we have an ANN with 4 inputs, 2 hidden layers with size 3 and 2, and 1 output. The forward-pass for the i-th sample is evaluated as follows:

$$z^{[1]} = W^{[1]}x^{(i)} + b^{[l]} \tag{3.7a}$$
$$a^{[1]} = \sigma(z^{[1]}) \tag{3.7b}$$
$$z^{[2]} = W^{[2]}a^{(1)} + b^{[2]} \tag{3.7c}$$
$$a^{[2]} = \sigma(z^{[2]}) \tag{3.7d}$$
$$z^{[3]} = W^{[3]}a^{(2)} + b^{[3]} \tag{3.7e}$$
$$\hat{y}^{(i)} = a^{[3]} = \sigma(z^{[3]}) \tag{3.7f}$$

The choice for the last activation function depends on the problem we are trying to solve: in case of a regression problem, we can use tanh, sigmoid or ReLU functions, while for classification problems, it is common to apply the softmax activation function, which outputs the probabilities of input to belong to the different classes.



(a) Sigmoid function and its derivative.

(b) Hyperbolic tangent function.

Figure 3.5: Two common used activation functions.

3.2.3 Backpropagation with Gradient Descent

In this section, we describe the loss and cost functions, and the backpropagation algorithm, following the notation given in many text books. Given the output $\hat{y}^{(i)}$, we define a *loss function* $\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$ as a function that measures the error from the real output $y^{(i)}$. In a real-valued regression problem it is common to choose $\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$ as the half-squared error loss:

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}||\hat{y} - y||^2 \tag{3.8}$$

while for binary classification using logistic regression the loss function normally is

$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y})) \tag{3.9}$$

The loss $\mathcal{L}^{(i)}$ outputs a scalar. Given this value, the training of the neural network consists of updating all the parameters $(W^{[l]}, b^{[l]})$ in the different layers. For any given layer l, using the *stochastic gradient descent* method, we update them as follow:

$$W^{[l]} = W^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial W^{[l]}} \tag{3.10a}$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial \mathcal{L}}{\partial b^{[l]}} \tag{3.10b}$$

where α is the learning rate.

Given m the dimension of the training set, we define the *cost function* $J(W, b)$ as the average of the m losses calculated over the m training samples:

$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)} \tag{3.11}$$

Using the *gradient descent* method, for any single layer l, the update rule is defined as:

$$W^{[l]} = W^{[l]} - \alpha \frac{\partial J}{\partial W^{[l]}} \tag{3.12}$$

The difference between the gradient descent (also called *batch gradient descent*) update versus the stochastic gradient descent version is that the cost function J gives more accurate gradients whereas $\mathcal{L}^{(i)}$ may be noisy. Stochastic gradient descent attempts to approximate the gradient from (full) gradient descent. The disadvantage of gradient descent is that it can be difficult to compute all activations for all examples in a single forward or backwards propagation phase. In practice, research and applications use mini-batch gradient descent. This is a compromise between gradient descent and stochastic gradient descent. In the case mini-batch gradient descent, the cost function J_{mb} is defined as follows:

$$J_{mb} = \frac{1}{B} \sum_{i}^{B} \mathcal{L}^{(i)} \tag{3.13}$$

where B is the number of examples in the mini-batch.

3.3 Recurrent Neural Networks

Recurrent neural networks or RNNs [99] are a family of neural networks for processing sequential data. In the last few years, they have been applied with success to a variety of problems: speech recognition, language modeling, translation, image captioning. RNNs can scale to long sequences and also they are able to process sequences of variable length. The typical representation of a RNN is given in Fig.3.6 and consists of a neural network with loops, that allow information to persist. In the diagram, a neural network, A, looks at some input x_t and outputs a value h_t:

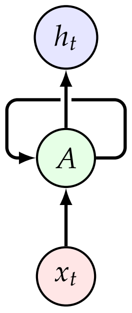


Figure 3.6: Rolled RNN structure.

RNNs can be thought of as multiple copies of the same network, each passing a message to a successor. The idea behind this architecture is to exploit the sequential structure of the data. The name of this neural networks comes from the fact that they operate in a recurrent way. This means that the same operation is performed for every element of a sequence, with its output depending on the current input, and the previous operations. This is achieved by looping an output of the network at time t with the input of the network at time $t+1$. These loops allow persistence of information from one time step to the next one.

The structure with loops in Fig. 3.6 becomes intuitive when we look at the chain formed when we unroll the computational graph. In the right part of Fig. 3.7, we

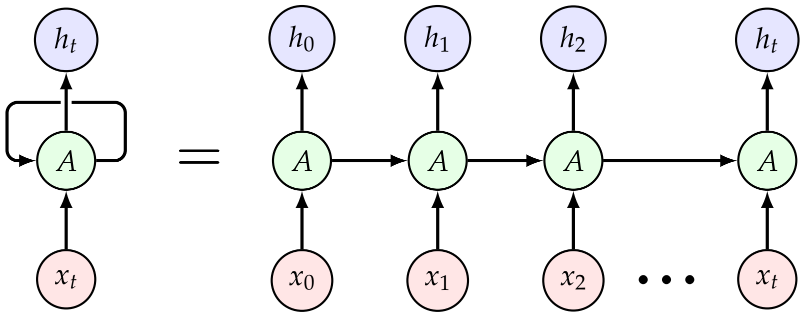


Figure 3.7: Unrolled RNN structure.

represent the same unrolled (or unfolded) network graph, where each node is associated with a particular time. Now we have an architecture which can receive different inputs at each time step x_t, has the capability of producing outputs at each time step h_t and maintains a memory state which contains information about what happened in the network up to time t.

3.3.1 LSTM Networks

To learn long-term dependencies, a solution consists to use deep recurrent neural networks. In theory, RNNs are absolutely capable of handling such long-term dependencies. However, in practice, when the number of layer increases, the training of network using the backpropagation algorithm can be difficult. In fact, what happens is that the gradient tends to get smaller as we move backward through the first hidden layers. This means that neurons in the earlier layers learn much more slowly than neurons in later layers. The problem was explored in depth by [100], and it is known as the *vanishing gradient problem*.

Long Short-Term Memory Networks are a particular kind of RNN, that have been introduced in [101]. They have been explicitly designed to avoid the vanishing-gradient problem in normal RNNs [102]. The capability of learning long-term dependencies is due to the structure of the LSTM units, which, differently to plain neurons, incorporates gates that regulate the learning process. The neurons in the hidden layers of an LSTM are Memory Cells (MCs). A MC has the ability to store or forget information about past network states by using structures called *gates*, which consist of a cascade of a neuron with sigmoidal activation function and a pointwise multiplication block. Thanks to this architecture, the output of each memory cell possibly depends on the entire sequence of past states, making LSTMs suitable for processing time series with long time dependencies [103].

A standard LSTM memory cell diagram is presented in Fig. 3.8. The basic operations are accomplished by the input gate i_t, the forget gate f_t and the output gate o_t:

- the *input gate* is a neuron with sigmoidal activation function (σ). Its output determines the fraction of the MC input that is fed to the cell state block;
- similarly, the *forget gate* processes the information that is recurrently fed back into the cell state block;
- the *output gate*, instead, determines the fraction of the cell state output that is to be used as output of the MC at each time step;
- moreover, the *cell state* c_t represents the memory of the unit and it is updated with the information to be kept (or to be forgotten), provided by the input gate (or forget gate).

Gate neurons usually have sigmoidal activation functions (σ), while the input and cell state use the hyperbolic tangent (tanh) activation function. All the internal connections of the MC have unitary weight [103]. With reference to Fig. 3.8, we report the operations performed by a single MC unit, at time t:

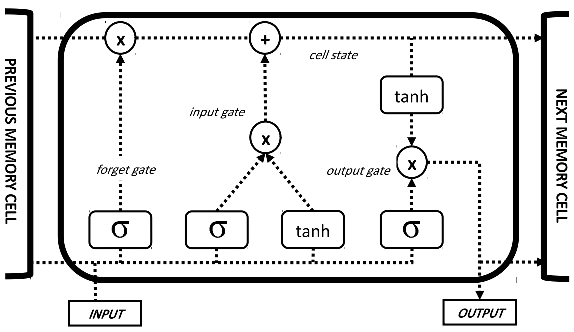


Figure 3.8: Standard LSTM unit.

$$i_t = \sigma(W_i \cdot (h_{t-1}, x_t) + b_i) \tag{3.14a}$$

$$f_t = \sigma(W_f \cdot (h_{t-1}, x_t) + b_f) \tag{3.14b}$$

$$o_t = \sigma(W_o \cdot (h_{t-1}, x_t) + b_o) \tag{3.14c}$$

$$\tilde{c}_t = \phi(W_c \cdot (h_{t-1}, x_t) + b_c) \tag{3.14d}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.14e}$$

$$h_t = o_t \odot \phi(c_t) \tag{3.14f}$$

In the previous equations, $\sigma(\cdot)$ is the sigmoid function and ϕ is hyperbolic tangent function (tanh). W and b are respectively the weight matrix and the bias of the gates i, f, o or of the cell state c. The subscript t is the time index and $\odot$ is the element-wise multiplication. The LSTM unit combines the output of the previous unit h_{t-1} with the current input x_t using the input, the output and the forget gates to update the memory of the cell. The variables i_t and f_t represent respectively the information that need to be kept or to be forgotten from the past and the current input. The cell state c_t is updated by summing the previous cell state c_{t-1} and the candidate cell state $\tilde{c}_t$, weighted respectively with f_t and i_t. Finally, we obtain the output h_t applying the tanh function to c_t and multiplying it by o_t. Then, the current output h_t is passed to next unit and combined with the input at the next time index $t + 1$.

3.4 Convolutional Neural Networks

3.4.1 Parameters Sharing

Convolutional Neural Networks (CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. To understand the differences from vanilla multi-layer networks to CNNs, we need to refer to the following concept: *parameter sharing* across different parts of a model. Sharing of the parameters makes it possible to extend and apply the model to examples of different forms (different lengths) and generalize across them. Such sharing is particularly important when a specific piece of information can occur at multiple positions within the input data. For example, in a object classification problem, the exact location of the object to be recognized in a given image is not known a-priori.

A traditional fully connected feed-forward network would have separate parameters for each input feature, so it would need to learn all the rules separately at each position. If we had separate parameters for each value of the input, we could not generalize to input that have not been seen during training, nor share statistical strength across different input lengths and across different positions of the input. By comparison, a convolutional neural network shares the same weights across several input location.

3.4.2 Basic CNN Structure

Convolutional Neural Networks differs from fully connected MLP for the presence of one or more *convolutional layers*. At each convolutional layer, a number of *kernels* (or *filters*) is used. Each kernel is composed of a number of weights and is convolved across the entire input signal to create multiple feature maps. In Fig. 3.9, we represent a typical CNN for image recognition.

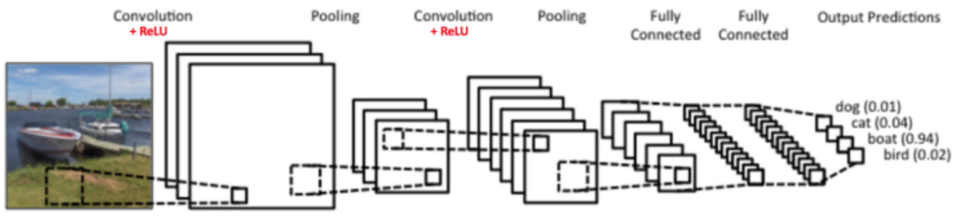


Figure 3.9: Typical CNN Structure.

The three feature operations that are computed by a CNN are:

- Convolution with a set of kernels;
- Non-Linearity Activation Function;
- Pooling or Sub Sampling

Note that the kernel acts as a filter, whose weights are re-used (shared weights) across the entire input: this makes the network connectivity structure sparse, i.e., a small set of parameters (the kernel weights) suffices to map the input into the output. With respect to other Neural Networks (e.g. LSTM Networks), this leads to a considerably reduced computational complexity with respect to fully connected feed forward neural networks, and to a smaller memory footprint. CNNs have been proven to be excellent feature extractors for images and inertial signals [104]. For more details the reader is referred to [105].

3.5 Software Framework

3.5.1 ML Libraries and Implementation

The success of ML and its countless applications is also due to the development of a multitude of frameworks that allows to implement and deploy ML models with relative easiness. In this thesis, the implementation of ML algorithms is done using open-source libraries: we choose Python as primary programming language, since, in the last couple of years, it became the main language for ML development and data science, and it can count on the most known up-to-date libraries for ML [106].

Moreover, we rely on cloud infrastructures for managing the collected network data and for the algorithms development. The reasons are manifold: first, there exist cloud services that allow to store data on the Internet for free, so it can be accessible everywhere. Second, most of the time these services integrate free GPU and Tensor Processing Unit (TPU) kernels, which are specifically designed to speed up the training of deep networks thanks to their parallel processing. Even if the free versions of the cloud services have some limitations in terms of storage and usage time, in most cases they are sufficient for our implementations. In Table 3.1, we report the main tools used in our development.

Name	**Type**	**Characteristics**
NumPy	Python Library	Math library that adds support for large, multi-dimensional arrays and matrices Includes a large collection of high-level mathematical functions to operate on arrays
Scikit-Learn	Python library	ML library that includes fundamental classification, regression and clustering algorithms Used for data preprocessing and for algorithms benchmark
Pandas	Python Library	Library for data manipulation and analysis It offers data structures and operations for manipulating numerical Library and time series. Used to load the data and for preprocessing
Matplotlib	Python library	Plotting library for NumPy It provides an object-oriented API for embedding plots into applications Designed to closely resemble that of MATLAB
Seaborn	Python Library	Python data visualization library based on matplotlib It provides a high-level interface for drawing attractive and informative statistical graphics
Tensorflow	Python Library	End-to-end platform for ML dataflow and differentiable programming across a range of tasks It has a comprehensive, flexible ecosystem of tools that lets researchers push the state-of-the-art in ML and developers build and deploy ML applications It is used for both research and production at Google

Keras	Python Library	Library for neural-network implementations It is capable of running on top of TensorFlow, R, Theano, and other ML backends Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
Jupyter Notebook	Dev Environment	Web-based interactive computational environment for Python Used for development with markdown texts, in-line plots and comments
Google Colab	Dev Environment	Google's free cloud service for AI developers similar to Jupyter Notebook Integrate main ML libraries and allows multi-user contributions Free GPU and TPU for neural networks training

Table 3.1: ML Frameworks and Libraries.

Chapter 4

Network Measurements and Dataset Collection

In this Chapter we present the methodology for the data collection that has been retrieved from operative Spanish mobile networks. This part represents a major contribution of the work and it is not limited to the applications presented in the next chapters of the thesis. Starting from Section 4.1, we discuss the motivations and the challenges to acquire this type of dataset, which includes radio level information at a time granularity not available in other studied datasets. Next, in Section 4.2, we explained the main characteristics of the LTE Physical Channel, from which we decode the data to obtain the network information. Finally, in Section 4.3, details of the measurement campaign is given.

4.1 Dataset with Radio Level Information

4.1.1 Motivations

Reliable network measurements are a fundamental component of networking research as they enable network analysis, system debugging, performance evaluation and optimization. However, setting up a network measurements campaign often require very expensive tools that are almost always exclusive prerogative of industries and mobile operators. In order to study the mobile networks from a comprehensive and meaningful dataset, we identify four different solutions and, for each, we individuate pro and cons which are summarized in Table 4.1

- **Simulated Data**: simulation data obtained though open-source network simulator (e.g. NS-3 [107]);
- **Operators Released Data**: datasets released by network operators (e.g. Telecom Big Data Challenge [28], see Section 2.1 for details);
- **Online Open-access Databases**: dataset freely available online (e.g. OpenCellId [15]);

- **New Data Collection**: gathering a complete new dataset from operative mobile networks to be studied.

Source	Pros	Cons	Examples
Simulated Data	Open-source solutions Relatively-short time to obtain the data Controlled environment	Not realistic, too simplified scenarios Cannot capture long-term traffic dynamics	NS-3 [107]
Operators Released Data	Realistic dataset From operative networks	Limited information and missing details privacy and open-access issues Coarse spatio-temporal aggregation	CDRs dataset [28]
Online Open-access Databases	Open-access information Fast to obtain large amount of worldwide data	Limited information and missing details Unverified sources and not comprehensive	OpenSignal Database [33]
New Data Collection	Realistic data and different scenarios Full control of time-space resolution Correlation with land-use and users activities	Time-consuming for collection and maintenance Measurements errors troubleshooting required	LTE Air Traffic Monitor [16]

Table 4.1: Source of available mobile network traffic datasets to be studied: pros and cons.

We weighted our options and possibilities based on multiple factors. First, we identified that it is important for us to control the *granularity* and the *resolution* of the data, disregarding the storage volume: in any data-driven problem, a large pool of data must be seen as an opportunity to obtain deeper insights and achieve an optimal solution of the problem, even if the complexity increases. This is more important when we consider ML solutions, for which the model outcomes highly depends on the quality and quantity of the analyzed data. Thanks to advancement in ML frameworks, most known ML algorithms are implemented and, therefore, the collection of a qualitative dataset represents a significant advancement for the analysis.

Considering that 4G works at milliseconds time-scale while 5G is characterized by even lower granularity, since it can allocate resources at the symbol level, the analysis of aggregated data with low frequency updates cannot help us to identify potential suboptimal performances. Moreover, another factor to be considered is that the dataset should give us a *network-side* perspective: obtaining data for a single customer would allow an improved user's profile characterization. However, we are more interested to understand how the network allocates the available radio-resources, if there is room for optimization and improvement, and the possible correlations between users' activity patterns and the land-use of the studied area.

These considerations are fundamental for the subsequent analysis and for the work objectives, and they lead us to consider the collection of new data from the operative mobile network. Moreover, relying on a new type of dataset would contribute to the state-of-the-art, since it allows to study the network starting from different source of information. The choice of which data to acquire considers also feasibility factors, including *costs*, *time* and *computational* efforts for the *collection*, *management* and *analysis* of the new dataset. The long-term objective also is not limited to the work of these thesis: this methodology is finally intended to obtain and an open-database with up-to-dates and ongoing measurements that can serve as baseline for the future research in mobile networks.

Due to the complexity and to the wide typology of information in mobile networks, many challenges arise when we start to consider a new network data collection. Big data technologies can help to deal with such problems, and they can answer to 4 Vs that characterize the plethora of information produced by modern communication systems: *Velocity* (frequency of data generation), *Variety* (heterogeneity of the data), *Volume* (total amount of data), *Veracity* (usability of the data).

When we start to deal with complex systems, such as 4G and 5G mobile networks, we understand that these 4Vs are not only theoretical principles, but real issues arise and they must be solved properly. For example, in our context we face the following cases:

1. many 4G operators work on 20 MHz bandwidth, and, using Carrier Aggregation (LTE CA), the total available bandwidth can be higher than 40 Mhz, enabling high throughput and large amount of data exchange (*Volume*);
2. each eNB can serve up to 1000 users (in case of Macro eNBs) and allows for speed up to order of Gb per second; the minimum periodic time frame for transmission in LTE (TTI) is in the order of ms (*Velocity*);
3. data are transmitted by diverse type of devices of different users, using different services and applications (*Variety*);
4. a large volume of data are dedicated to planning, control and maintenance of the network functionalities (e.g. scheduling information): is it all the exchanged information useful for the network analysis (*Veracity*)?

The previous example refer to the 4G technology. With 5G, the network performances are promised to be boosted by at least 10x factor. Moreover, we do not account for the myriad of promised applications and services (e.g. autonomous driving, augmented and virtual reality, IoT devices), which will contribute to increase the overall complexity of the network.

4.2 LTE Physical Control Channel

In LTE, the Physical Control Channel is responsible for the exchange of scheduling information between the LTE Base station (eNodeB) and User Equipment (UE). Due to the separation between Data Plan and Control Plan in LTE, no personal data are exchanged over the Control Channel. The users' actual data exchanged between the network are sent over encrypted channel (Physical Downlink/Uplink Shared Channel, PDSCH/PUSCH). Instead, the Physical Downlink Control Channel (PDCCH) is unencrypted but the subscribers privacy is preserved, since the users' identities are obfuscated using temporal identifiers that are renewed after short periods of inactivity. Decoding the LTE control channel would give access to the full base station traffic at a TTI granularity, which is 1 ms in LTE, thus allowing for traffic profiling and accurate measurements. Although few implementations of LTE PDCCH decoders are available [16], not all of them are free or they do not provide tools to reliably decoding the LTE control channel and, thus, accessing the scheduling information.

4.2.1 OWL: Online Watcher for LTE

In [16], the authors present OWL, an Online Watcher for LTE that is able to decode the resource blocks in more than 99% of the LTE frames, significantly outperforming existing non-commercial decoders. Compared to previous attempts, OWL makes possible to run the software on inexpensive hardware coupled with almost any software defined radio capable of sampling the LTE signal with sufficient accuracy. This tool has been specifically designed for researchers that need an economic solution to perform reliable measurements on LTE physical communications between mobile phones and the serving base station. OWL is built on top of srs-LTE, an LTE library that provides an efficient implementations of LTE physical channels and works with a few software defined radios (SDRs), including bladeRF and USRP, which are capable of sampling LTE signal. In particular, [16] extends srs-LTE by implementing an online procedure to decode all Downlink Control Information (DCI) transmitted on the PDCCH. This solution is more efficient than previous attempts, because the software collect and maintain a list of active Radio Network Temporary Identifiers (RNTIs), which identify user equipments (UEs) within a given cell (eNodeB).

The OWL software architecture is composed of three processes: 1) a synchronized signal recorder, 2) the actual control channel decoder, and 3) a fine-tuner that is used when a control message is expected to be found on the control channel, but the main process cannot decode it. Finally, an auxiliary verifier tool checks whether the decoded DCIs match the actual resource allocation on the PDSCH.

1. **Synchronized signal recorder:** OWL first, synchronizes the software to the eNodeB transmissions by means of Primary Synchronization Sequence (PSS) and Secondary Synchronization Sequence (SSS) correlation, then acquires the remaining information by decoding the MIB, and finally it writes an output

file starting from the first symbol of the first frames for which it obtained a successful MIB decoding. However, it might happen that the system synchronization degrades without the recorder being able to notice, in particular for recordings longer than a few seconds. To improve this, the recorder performs synchronization check at the beginning of every frame. In addition, an error log tracks any synchronization issues and any other software related error that might hamper the following operations.

2. **Control channel decoder** OWL's main component is the control channel decoder. It can work either online while the signal is being sampled by the SDR or offline processing prerecorded traces. While a single UE can monitor a limited set of control channel locations, OWL needs to extend the procedure to all possible locations and DCI formats. OWL only performs actual DCI decoding if there is an ongoing transmission on the resource elements of the scanned location. With respect to srs-LTE, OWL considers the decoding operation successful if any of the C-RNTI of the active list matches with the decoded message. Since the C-RNTI list is empty when the system starts, OWL needs to populate it while decoding the control channel.

   To do so, OWL can either 1) exploit the random access procedure or 2) verify the decoding success by re-encoding the DCI. In the former procedure, whenever a DCI is decoded with the Cyclic Redundancy Check (CRC) field XORed with a Random Access RNTI (in [1-10]), not only is it considered a successful decoding, but also the Random Access Response (RAR) message, which is sent in PDSCH, is actually demodulated and decoded and provides OWL with a new C-RNTI to be inserted in the active list. The particular configuration of the RAR messages allow us to simplify the decoding by just taking the last two bytes of the message, because the C-RNTI is always specified in this location. In addition, since the actual RAR message is provided with a CRC field, OWL is able to evaluate the correctness of the whole operation by verifying the message checksum against the CRC field.

   The RNTIs are just temporary identifiers and, after a complete SFN cycle (10.24 seconds) of inactivity, a UE needs to perform the access procedure again to obtain a new one. For this reason, OWL resets all the RNTIs in the list that are inactive for more than a SFN cycle.

3. **Fine-tuner and Verifier** While, theoretically, the control channel decoder should be able to decode all DCIs, there are a few rare conditions for which power is detected on the control channel, but no DCI message has been decoded. Authors of [16] believe that these conditions are due to either equalization or synchronization problems. The fine-tuner is able to correct the majority of these issues by iteratively performing the decoding operation on the specific location only and varying the timing offset of the LTE signal.

Finally, to verify whether the decoded information matches the actual PDSCH resource allocation, a simple tool takes as inputs the decoding log and the raw LTE signal trace. For each subframes it computes how many RBs are detected by OWL by summing all the NRB values of downlink messages. Similarly, it evaluates for each subframe and for each RB whether the average power measured on the PDSCH is higher or lower than the power measured on the reference signals that are the closest to the related RB.

4.2.2 DCI Decoding

Decoded and verified DCI messages include the scheduling information for the connected User Equipments (UEs) to a specific eNodeB. Among other information, DCI contains the following fields:

- **Radio Network Temporary Identifier (RNTI)**: DCI messages use RNTIs to specify their destination. RNTIs are 16-bit identifiers that are employed to address UEs in an LTE cell. They are used for different purposes such as to broadcast system information (SI-RNTI), to page a specific UE (P-RNTI), to carry out a random access procedure (RA-RNTI), and to identify a connected user (C-RNTI). Here, we are interested in the C-RNTI, that is temporarily assigned when the UE is in RRC (Radio Resource Control) CONNECTED state, to uniquely identify it inside the cell. The C-RNTI can take any unreserved value in the range [`0x003D–FFF3`]. ;

- **Resource Block (RB)** assignment: in LTE, a RB represents the smallest physical resource unit in time and frequency that can be allocated to any user. The number of resource blocks that are assigned to a UE (N_{RB}), is derived based on the DCI bitmap;

- **Modulation and Coding Scheme (MCS)**: the MCS is a 5-bit field that determines the modulation order and the code rate that are used, at the physical layer, for the transmission of data to the UE;

Based on RB and MCS, we can calculate the **Transport Block Size (TBS)**. The TBS specifies the length of the packet to be sent to the UE in the current Transmission Time Interval (TTI). It is derived by from a lookup table by using MCS and N_{RB}, see [108].

4.3 Measurements Campaign

4.3.1 Scenarios and Objectives

The measurement campaign has been planned with the objective of collecting an exhaustive dataset that comprehends different scenarios, in order to be able to describe as better as possible most of the potential realistic use-cases and to be usable for the applications presented in the rest of this thesis. As guideline, the 3GPP standard release technical reports (e.g. [109], see Table 4.2), where a set of use-cases, and their correspondent requirements, are listed. For example, urban locations require more capacity per square meters, and therefore the operators are called to plan the network deployment in such a way to satisfy the specified requirements.

Scenario	Experienced data rate (DL)	Experienced data rate (UL)	Area traffic capacity (DL)	Area traffic capacity (UL)	User density	Activity factor
Urban macro	50 Mbps	25 Mbps	100 Gbps/km^2	50 Gbps/km ^{2}	10 000/km^2	20%
Rural macro	50 Mbps	25 Mbps	1 Gbps/km^2	500 Mbps/km^2	100/km^2	20%
Indoor hotspot	1 Gbps	500 Mbps	15 Tbps/km^2	2 Tbps/km^2	250000/km^2	N/A
Dense urban	300 Mbps	50 Mbps	750 Gbps/km^2	125 Gbps/km^2	25000/km^2	10%
Highspeed train	50 Mbps	25 Mbps	15 Gbps/train	7.5 Gbps/train	1000/train	30%
Highspeed vehicle	50 Mbps	25 Mbps	100 Gbps/km^2	50 Gbps/km^2	4 000/km^2	50%
Airplanes connectivity	15 Mbps	7.5 Mbps	1.2 Gbps/plane	600 Mbps/plane	400/plane	20%

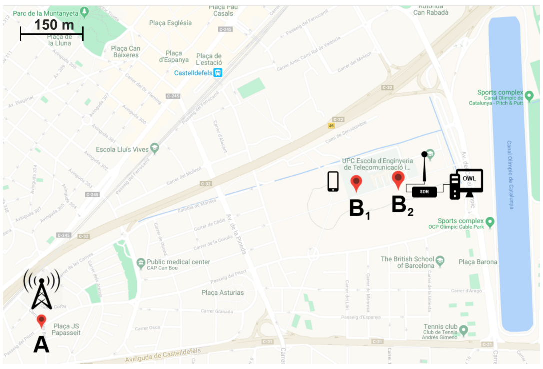
Table 4.2: 3GPP Technical Report 22.261: Mobility Use Cases [109].

Keeping this in mind, we placed our LTE sniffer in different areas characterized by different land-use and, consequently, different users' behaviours. One of the main objective is to have mobile traffic network that is generated and transmitted both in urban and sub-urban area. Also, we want to characterize the diversity of network traffic, for example, between office and residential area, where the traffic profiles show different characteristics both in term of magnitude, shape and demanded traffic type. The variability of the traffic is also a fundamental feature that we want to characterize: to this end, we place the sniffer in areas where the traffic shows a cyclic weekly behaviour, but we monitored also different locations where the traffic varies in sudden and unexpected ways.

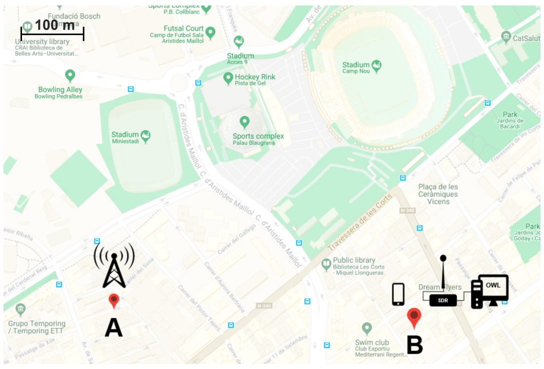
4.3.2 Setting and Measurement Scheduling

We planned to collect our dataset from different locations of the city of Barcelona, Spain and we extend it with additional data retrieved from another measurement campaign that took place in Madrid, thanks to Nicola Bui, who originally proposed this for data collection. Therefore, our overall dataset contains mobile traffic information for the two major cities in Spain: Barcelona and Madrid.

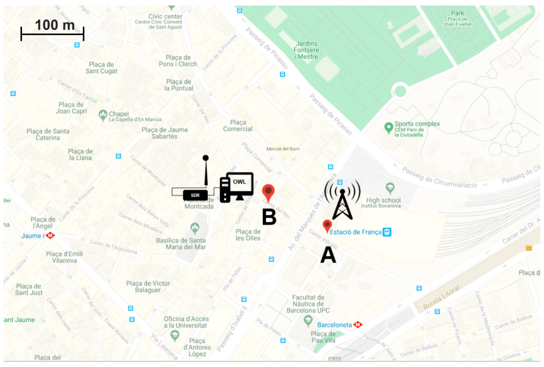
In Table 4.3, we report the locations where the sniffer has been placed and the corresponding land-use of that area. In Fig. 4.1, we show the locations of the sniffer and the monitored cell in few areas in Barcelona. During the measurement period, we are able to remotely monitor the recording activity thanks to web-based application that simply plot the downlink and uplink traffic measured at that cell. The app is shown in the monitor in Fig. 4.3, and a snapshot is reported below.



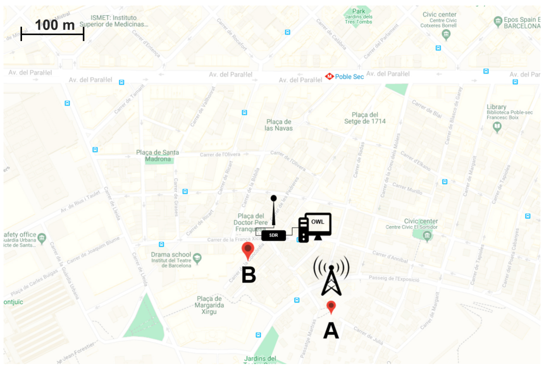
(a) Castelldefels: suburban area with a university campus.



(b) Camp Nou: mainly residential area with Barcelona FC stadium.



(c) Born: mixed residential, transport and leisure area.



(d) PobleSec: mainly residential area.

Figure 4.1: Maps of Barcelona metropolitan areas where the measurement campaign took place. The eNodeB location is denoted by A, the sniffer is marked as B.

A representation of the setting for the sniffer is given in Fig. 4.3. It consists of three parts (plus a UE terminal):

- **SDR**: the LTE DCI decoding software supports a wide range of SDR; as suggested by the author, we use the Nuand BladeRF x40 SDR;
- **PC**: in our case, we choose Intel mini-NUC, equipped with an i5 2.7 Ghz multicore processor, 256 GB SSD storage; due to its reduced size it represents a good compromise between computational power and portability; and 16 GB of RAM.

Cell Name	City	Approximate Location	Land-use Characteristics
atocha	Madrid	Atocha Train Station	Train station area
callao	Madrid	Callao Square	One of the main squares of the city; along the path of the most central shopping and restaurant street
imdea	Madrid	Leganes	Office and residential area
leganes	Madrid	Leganes	Mainly residential area with a few commercial activities in the surroundings
rastro	Madrid	Rastro Market	Location known for the most popular market in the city
born	Barcelona	Born District	Mixed residential, transport and leisure area
campnou	Barcelona	Camp Nou Area	Mainly residential area with Barcelona FC stadium
castelldefels	Barcelona	Castelldefels	suburban area with a university campus
poblesec	Barcelona	Poble Sec Neighbourhood	Mainly residential
sants	Barcelona	Sants Train Station	Train station area

Table 4.3: Locations and land-use characteristics of the measurement campaign.

- **Antenna**: here, there is no strict specifications; so, one can use any antenna that supports the LTE frequencies

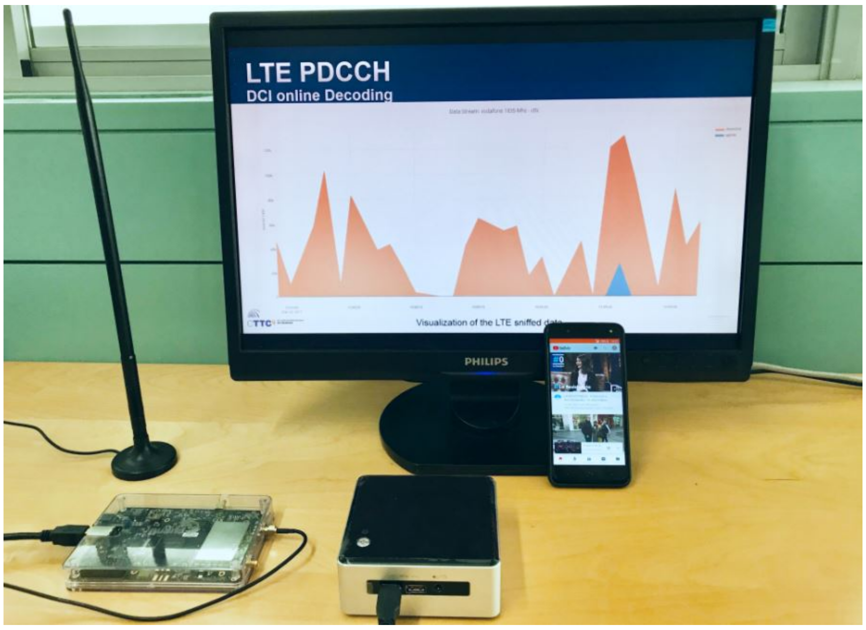


Figure 4.2: Experimental setup for data collection: a SDR module is connected to an antenna to capture PDCCH DCI data, which is decoded by a mini-PC running a dedicated software.

The screen monitor is not necessary for the duration of the measurements. Therefore, the sniffer overall sizes make it very compact and easily transportable. The setup is simple and requires to find the specific which is the Physical Cell ID (PCI) of the eNodeB to be monitored with OWL. This can be achieved also using a mobile terminal and through specific mobile applications it is possible to find the details of the surrounding cells, including the PCI. To test the synchronization between the sniffer and the cell, there exist some application that are part of the srs-LTE library [110] (see details in [16]). Moreover, using the website AntenaGSM.com, it is possible to know the exact position of the cell.

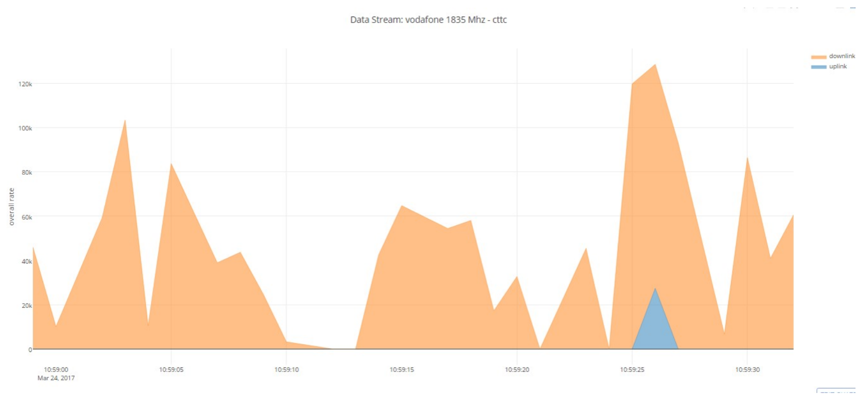


Figure 4.3: Snapshot of the web-app for real-time visualization of the collected LTE data.

Chapter 5

Mobile Traffic Characterization and Prediction

In this Chapter we present the analysis of real mobile traffic traces, which is helpful to understand usage patterns of cellular networks. In particular, mobile data may be used for network optimization and management in terms of radio resources, network planning, energy saving, for instance. First, in Section 5.1 we perform an analysis on the retrieved LTE raw traces: we give a temporal characterization of the traffic using different time-scales and we model it through Markov chains. This section is the result of the following publication:

- Trinh, H. D., Bui, N., Widmer, J., Giupponi, L., & Dini, P. (2017, October). **Analysis and modeling of mobile traffic using real traces**. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (pp. 1-6). IEEE.* [19];

In Section 5.2, we continue the study using the data collected in a residential and a campus area. We use an unsupervised method to classify the traffic sessions in clusters. This section is the result of the following publication:

- Rago, A., Piro, G., Trinh, H. D., Boggia, G., & Dini, P. (2019, June). **Unveiling Radio Resource Utilization Dynamics of Mobile Traffic through Unsupervised Learning**. In *2019 Network Traffic Measurement and Analysis Conference (TMA) (pp. 209-214). IEEE.* [20];

Finally, in Section 5.3, we perform mobile traffic prediction using a method that comprises LSTM neural networks. This section is the result of the following publication:

- Trinh, H. D., Giupponi, L., & Dini, P. (2018, September). **Mobile traffic prediction from raw data using LSTM networks**. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) (pp. 1827-1832). IEEE.* [21];

5.1 Analysis and Modeling using Raw LTE Traces

In the context of intelligent mobile networks, information about the users' traffic profiles and on the network usage patterns becomes essential during the phases of planning and of deployment of the network. This can translate into a more efficient allocation of the resources and can help mitigate the effects of the increasing costs incurred by the network operators to tackle the expected upsurge of the Internet demands.

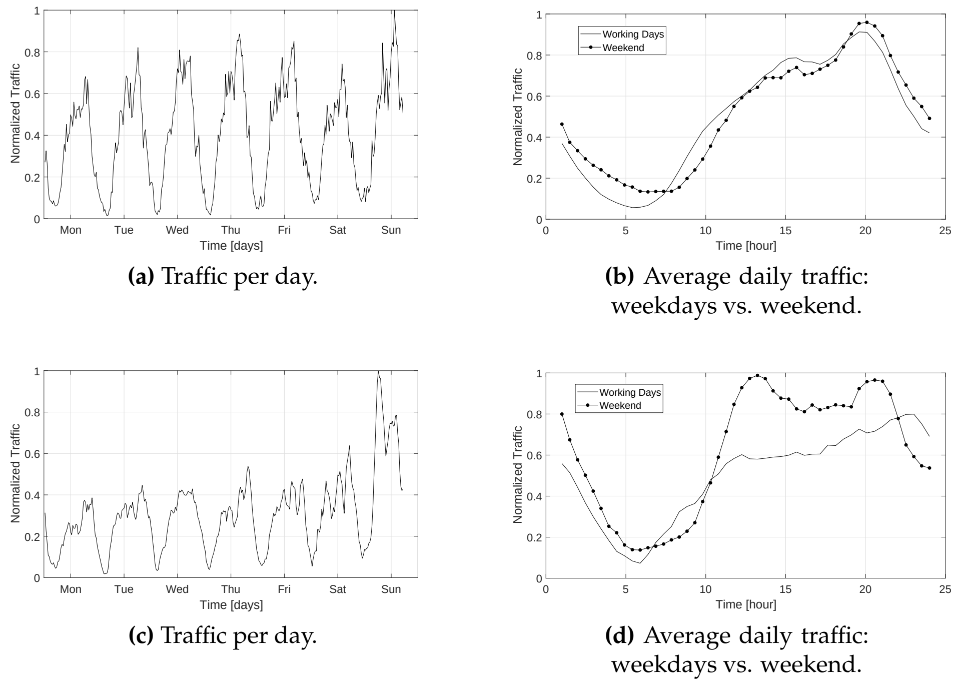
However, for research and academic communities, it is very challenging to get access to real data extracted from mobile network. We overcome the lack of network information using the LTE sniffer presented in the previous section, which is capable of decoding the unencrypted LTE control channel, and we present a temporal and spatial analysis of the recorded traces. Moreover, we present a methodology to derive a stochastic characterization for the daily variation of the LTE traffic. The proposed model is based on a discrete-time Markov chain and is compared with the real traces. Results show that, with a limited number of states, our model presents a high level of accuracy in terms of first and second order statistics.

5.1.1 Dataset Analysis

We derive our analytical model from an extensive dataset of LTE scheduling information, which have been collected in four locations of the metropolitan city of Madrid. In particular, the dataset has been collected using OWL [16], using a Software Defined Radio (SDR) to send the raw LTE signal to a PC running the decoding software. This open-source software is capable of reliably logging the LTE DCI broadcast by base stations [111].

Resources are assigned to devices through RNTIs, every millisecond, specifying the number of resource blocks (RBs) and the MCS index to be used. This makes our dataset both anonymous, because it is impossible to obtain users' unique identifiers, and accurate, because we can separate the dataset into high-resolution traces belonging to individual communications. Therefore, our datasets are useful to obtain both aggregated information on a given cell and to extract trace-based statistic distributions.

The analysis aims at describing the main characteristics of the LTE traffic by analyzing the number of connected users and, both, temporal and spatial variations of the collected traces. The results that we show refer to the downlink communication between the eNodeB and the user equipments. The traffic is normalized with respect to the peak traffic that occurred in the examined period. Without loss of generality, the same analysis can be extended to the uplink direction.



(a) Traffic per day.

(b) Average daily traffic: weekdays vs. weekend.

(c) Traffic per day.

(d) Average daily traffic: weekdays vs. weekend.

Figure 5.1: Weekly and daily traffic for two LTE eNodeBs with a time resolution of 30 minutes.

Temporal Behaviour

Figure 5.1 shows the downlink aggregated throughput of two eNodeBs averaged over the 30 days of monitoring. We can distinguish the traffic per week in Figure 5.1a and 5.1c and the daily traffic in Figure 5.1b and 5.1d.

A strong relation between mobile traffic and connected users is recognized. We observe the same daily pattern repetition: high traffic is shown during the hours of the day (when population is active), whereas less intensive traffic is experienced during nights (when people sleep). Traffic intensity is similar in working days and during weekends. A different behavior is detected in one particular cell, where a higher traffic is normally experienced on Sunday (Figure 5.1c). The reason behind this higher activity, is the presence of a local market open every Sunday in the same area where the eNB is located. As for the connected users, the minimum traffic is around 5.30 am for all the cells; a more prominent peak can be seen around at 8 pm. The maximum ratio between the peaks observed in the measurements is 13.3. The absolute values of the traffic are different and depend on the location of the area where the eNodeB is deployed.

Figure 5.2 shows the normalized average daily traffic distribution of the observed cells. Also in this case, the daily average traffic profiles are similar in shape for all the cells, especially during low load period. As a proof of representiveness of our measurements, we compare the extracted information from the collected data with the traffic model presented in the EU ICT FP7 EARTH [112]. The data used in this project are provided by a network operator. We observe that, considering a daily average, the two traffic shapes are compatible and very similar (see Fig. 5.2). This comparison does not account for the absolute values of the traffic, which are dependant on the location of the specific base station, but it shows, on average, how the traffic demand is distributed over 24 hours. A different coefficient for the traffic magnitude can be calculated for each eNodeB based on the active population of that zone.

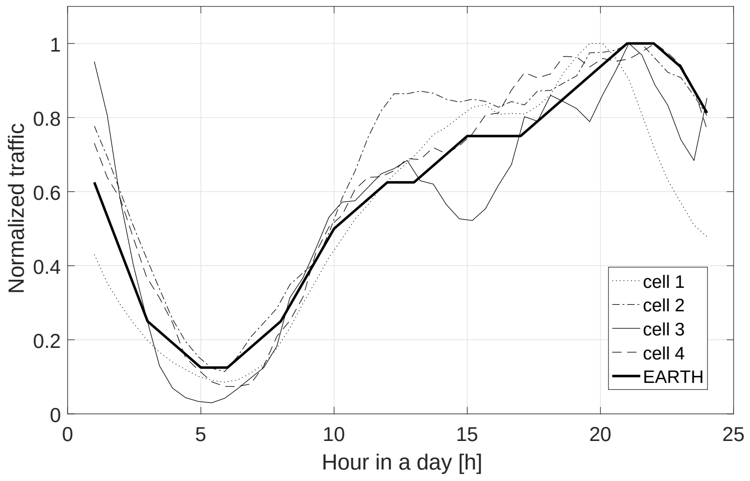
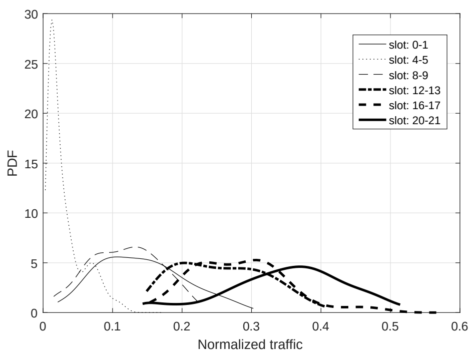


Figure 5.2: Normalized daily traffic of different LTE cells and EARTH model.

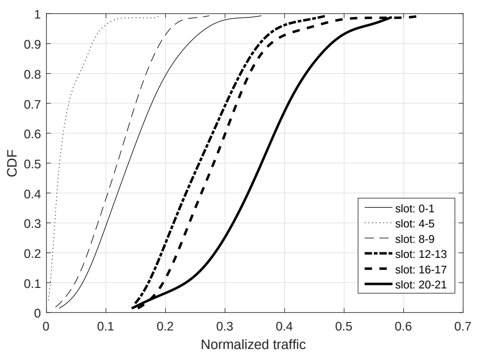
Next, we show some example statistics for the traffic intensity of the four observed cells. Figure 5.3 shows the probability density function (pdf) and the cumulative distribution function (cdf) by applying the Kernel Smoothing algorithm on the empirical data traces. We have computed the pdfs and cdfs for different periods of one day (slots of 1 hour duration) and we have also evaluated their variation during the day. The figure shows only 6 slots for the sake of simplicity. The numbers report the start/end hour of the day of the respective slot.

We notice that night and early morning are the periods with lower traffic intensity (slot 0-1 and slot 4-5 have curves on the left side of the graph). After that and till slot 20-21, the traffic is increasing (the curves are more on the right side of the x axis). Moreover, we can identify that the curves for slot 8-9 and slot 12-13 are similar, which indicates that the traffic in those hours is almost at the same level.

Finally, Figure 5.4a shows the number of connected users (both idle and active) in a cell during a day. The number is strictly correlated with the location where the e NodeB is deployed. In fact, cell 1 presents a higher number of users with respect to the others because it is deployed in the centre of the city with a high population density and activity. However, normalizing the curves with respect to the daily maximum number of users, the same pattern is identified for all the cells (Figure 5.4b).



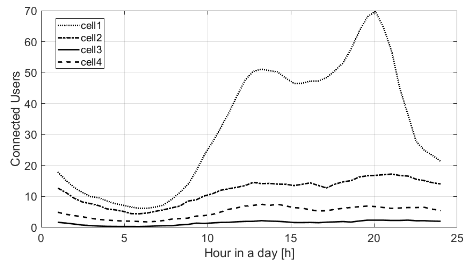
(a) PDF



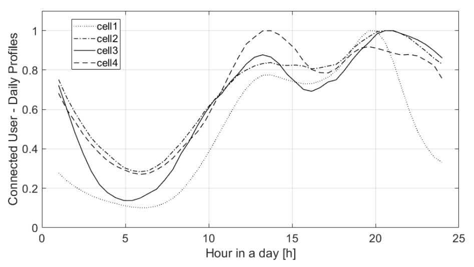
(b) CDF

Figure 5.3: PDF and CDF of the eNodeB traffic for six 1 hour-duration time-slots of a day.

The identified pattern follows a very similar behavior of the traffic profile. This confirms the correlation between the number of users and their generated traffic with the daily human activity.



(a) Absolute activity



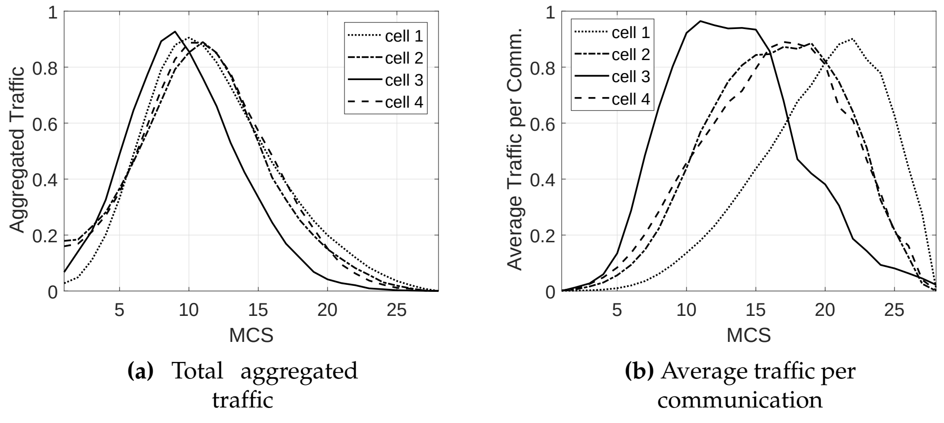
(b) Normalized activity.

Figure 5.4: Connected users per cell - comparison.

Spatial Behaviour

We are able to estimate the quality of the channel experienced by the users during the communication with the eNodeB, based on the assigned MCS index. One of the 28 possible MCS indexes is allocated by the eNodeB as a function of the Channel Quality Indicator (CQI) sent by the UE. The CQI depends on the SINR experienced by the user, which, among other factors, generally decreases with the distance between the eNodeB and the UE. In [113] a mapping between SINR values and different CQIs is provided. As a result of that, and based on the information on the assigned MCS, we estimate a spatial distribution of the user and combine it with the served traffic, in order to obtain a traffic distribution in space for each eNodeB.

Considering all the communications occurred in the recording period, Figure 5.5a shows the aggregated amount of traffic for each assigned MCS index: the top three indexes are 9, 10 and 11 and this is confirmed for all the analyzed base stations. On the other hand, we see different profiles (Fig. 5.5b when we consider only the average traffic per communication. This is due to the fact that the MCS indexes assigned by the eNodeB among the users are not uniformly distributed. For cell 1, except for the highest 3 MCS indexes, the users that experience a better quality of the channel also produce larger amount of traffic on average. However, a different behavior is noticed for cell 3: here, the largest communications correspond to a MCS index between 10 and 15.



(a) Total aggregated traffic

(b) Average traffic per communication

Figure 5.5: Traffic associated to different MCS (aggregated vs. average).

In Figure 5.6, we analyze more than 10 millions communication traces between the eNodeB and the users. This map shows the spatial distribution of the users' communications and the relative amount of traffic. Considering a cell in the center of the plot, the distance between the users and the eNodeB is distributed according to the average MCS experienced during the communication. The exact angular position of the user is unknown and it is picked from a uniform distribution. The total amount of traffic produced during the communication gives the magnitude, represented by the different traffic intensities in the figure. The contour lines in the map group the areas with similar traffic distribution and highlight those that produce the larger amount of traffic. The groups shown in the figure demonstrate that the central region of a cell is usually the most dense and produces most of the traffic.

5.1.2 Discrete-Time Markov Model

The proposed model aims at profiling the traffic pattern of a cell during a day. The daily time-scale has been selected based on the study of the frequency domain shown in Figure 5.7, which reports a strong periodicity of the traffic during the 24 hours.

The dynamics of the mobile traffic intensity are captured by a discrete-time Markov chain with N_s states. Formally, we consider a traffic intensity in bit per second during a given hour of the day, which can be in any of the states $x_s \in \mathcal{S} = \{0, 1, ..., N_s - 1\}$. Every time step, the system evolves from a state $x_s(k)$ to the next state $x_s(k+1) \in \mathcal{S}$

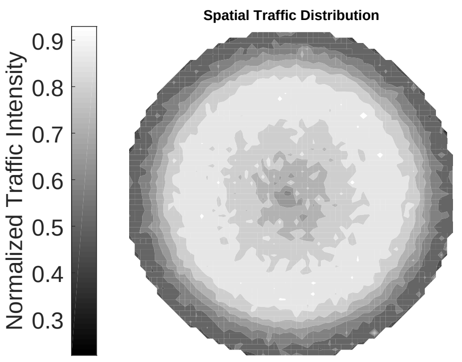


Figure 5.6: Spatial traffic distribution of more than 10 million traces for a single eNodeB.

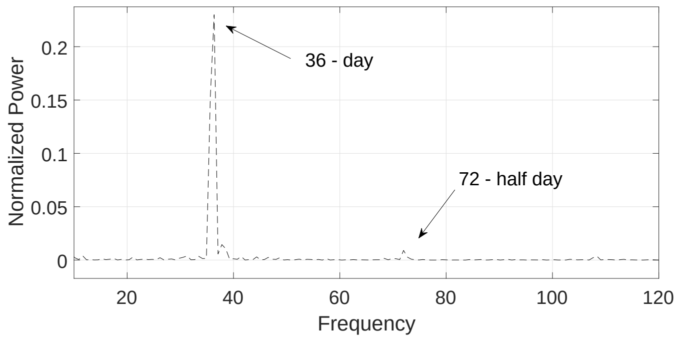


Figure 5.7: Periodogram of a 36 days-long traffic trace.

according to the probabilities $p_{uv} = Prob[x_s(k+1) = v|x_s(k) = u]$, with $u, v \in \mathcal{S}$, which is not null only if $t_{k+1} = (t_k + 1) mod N_t$, being N_t the number of time slots in a day. To calculate the one-step transition probabilities from empirical data, we use Algorithm 1: for each step, the algorithm computes the transition probability matrix by counting how many times the cell traffic moves from a state to another. We obtain the correspondent probability matrix by normalizing each row.

Algorithm 1 Transition Probability Matrix Calculation

```
1: procedure MARKOV MATRIX($data, N_t, N_s$)
2:     $qData \leftarrow$ quantize $data$ in $N_s$ levels
3:     for $t_s$ in $[0, ..., N_t - 1]$ do
4:         for $x_1$ in $[0, ..., N_s - 1]$ do
5:             for $x_2$ in $[0, ..., N_s - 1]$ do
6:                 $M_{x_1,x_2,t_s} \leftarrow$ count # transitions $x_{s1} \rightarrow x_2$ in $qData(t_s)$
7:     normalize rows of $M$
8:     return $M$
```

Model Results

In this section, we show some results on the stochastic Markov model for the daily traffic intensity. To evaluate our model, we split the dataset of a given cell into a training set and a validation set. The training set comprises 75% of the recording days and it is used to obtain the model through the presented algorithm. The validation set is used to have a numerical comparison with the traffic generated with the model.

Figure 5.8 shows the error due to the selection of the number of states N_s and the number of slots N_t. We apply a uniform quantization strategy that achieves accurate results, as demonstrated next. The error is calculated with respect to the validation trace as the average absolute daily difference, given by the following equation:

$$E_{rr} = \frac{1}{N_s} \sum_{i=0}^{N_s-1} |x_{sim} - x_{val}| \tag{5.1}$$

We notice that an increase in N_s and N_t corresponds to a decrease of the error. In particular, with $N_s \geq 6$ states and $N_t \geq 24$ time-slots, the error is small enough to produce a good approximation of the mobile traffic.

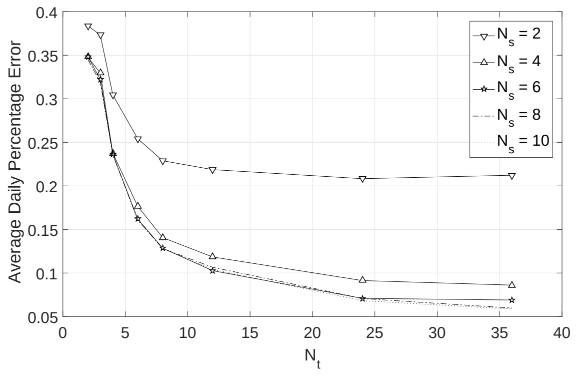
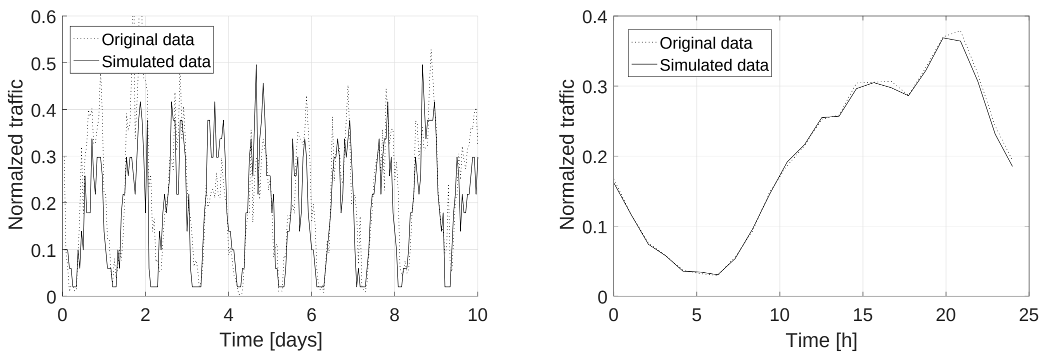


Figure 5.8: Error experienced changing the number of states N_s and the number of time-slots N_t.

Figure 5.9a shows a 10-days synthetic traffic trace versus the validation dataset and their daily average, using $N_s = 10$ and $N_t = 24$. We can see that, considering a sufficient number of days, the model is able to estimate with high accuracy the daily traffic pattern (Fig. 5.9b).

Considering one single cell, Fig. 5.10 demonstrates the statistical accuracy of the discrete-time Markov traffic model. It shows the cdf of the synthetic traces applying the Kernel-Smoothing algorithm with the cdf of the traces from the validation set. We observe that the two curves almost overlap. Kolmogorov-Smirnov test is passed with a confidence of 1%.

Moreover, we show that our Markov model is sufficient to accurately represent second-order statistic. Fig. 5.11 shows the autocorrelation function (ACF) for different values of N_s. With only 2-states ($N_s = 2$) the model is able to capture the



(a) Ten days. **(b)** Daily time scale.

Figure 5.9: Simulated Traffic vs Original Data with $N_t = 24$ and $N_s = 10$.

periodicity of the traffic profiles and classify it in high or low load periods. However, major accuracy requires higher values of N_s. With $N_s = 10$ the model already performs satisfactorily. The good fit of the autocorrelation function confirms that, for a sufficient value of N_s, a further of level complexity is unnecessary in the characterization.

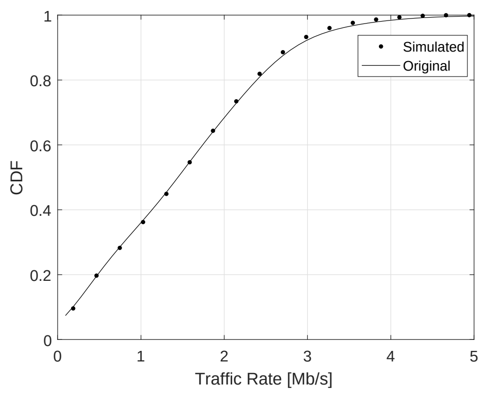


Figure 5.10: CDFs of the synthetic traffic trace vs empirical traces.

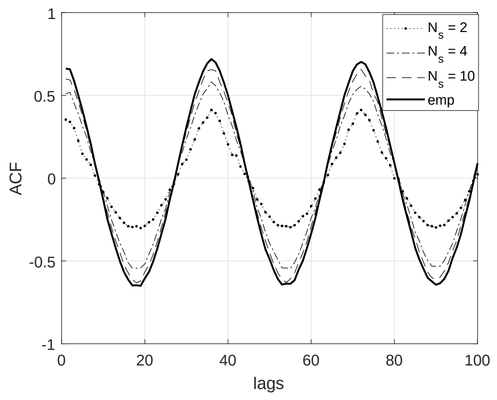


Figure 5.11: Autocorrelation function for the simulated traces and the empirical data for different values of N_s.

5.2 Unveiling Radio Resource Utilization Dynamics through Unsupervised Learning

In this Section, we continue the statistical analysis on part of the collected dataset, using an unsupervised learning method, namely K-means [114]. Specifically, a multivariate analysis on the mobile traffic sessions has been conceived to make a classification at the radio link level, according to their properties. To this end, K-means is used to map sessions with similar properties into K clusters.

5.2.1 Mobile Traffic Analysis and Discussion

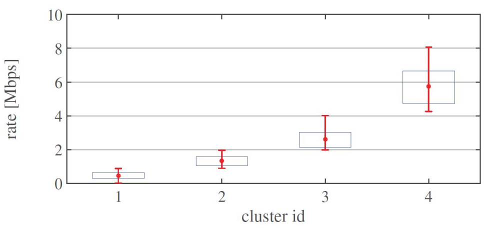
The study focuses on four features in the downlink communications to describe the mobile traffic sessions. They are the following: *average data rate*, *average MCS index*, *average number of allocated RBs*, *session duration*. The variables of interest are firstly normalized within the range]0,1]. Then, each session is represented as a point in a *hyperplane*, whose dimensions refer to the variables of interest of the conducted analysis. At this point, the dissimilarity associated with two sessions is defined as the Euclidean distance between the two related points in the aforementioned hyperplane. Indeed, the optimal value of K is calculated in order to ensure that the intra-cluster distances are minimized and the inter-cluster distances are maximized [114] (note that, according to K-means terminology, this means that the silhouette [115] is maximized). Finally, the clustering process provides in output the sessions of each cluster and a special point of the hyperplane, namely **centroid**, that identifies the cluster itself. Their coordinates are obtained by averaging the value of variables associated with the sessions belonging to the considered cluster. By studying the obtained groups of sessions, it is possible to extract statistical details associated with each cluster and finalize the traffic characterization.

Two LTE base stations in a residential and campus area of Barcelona, operating in a bandwidth of 20 MHz, are monitored to collect mobile data. The residential area dataset contains 521 sessions. Instead, the campus area dataset contains 4946 sessions. The analysis proposed next discusses the properties of the gathered mobile data for each base station considering (i) the dataset as a whole and (ii) the dataset divided into 4 time-slots that are morning, afternoon, evening, and night.

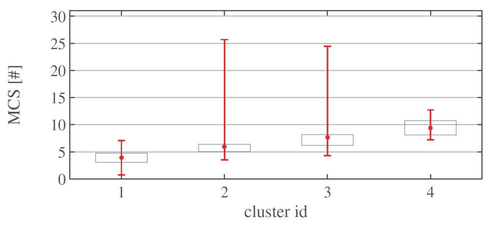
5.2.2 Study of the dataset as whole

The two monitored base stations show different behavior in terms of radio resource usage patterns. The first difference refers to the output of the silhouette analysis, which groups the residential and campus traffic in four and two clusters, respectively. Figures 5.12 and 5.13 show the outcome of the K-means clustering process, carried out for the residential area and the campus area. For each variable of interest, the figures highlight the identified clusters, their centroids (i.e., the red dots), the

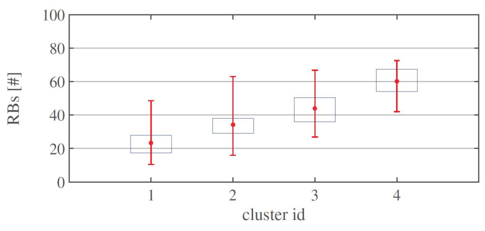
25th and the 75th percentile (i.e., the bottom line and the top line of the blue rectangle), as well as the minimum and the maximum measured value (i.e., the edges of the vertical red line) of the variables of interest.



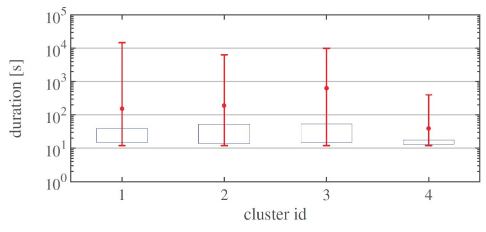
(a) Rate



(b) MCS

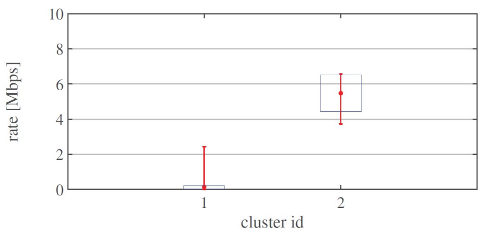


(c) RBS

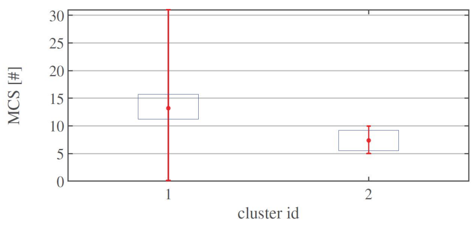


(d) Duration

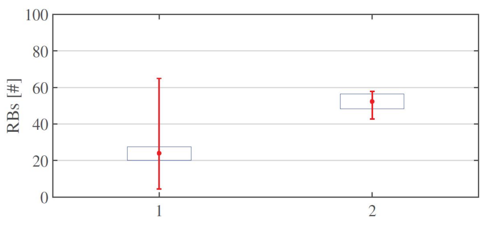
Figure 5.12: Study of dataset related to the residential area, as a whole.



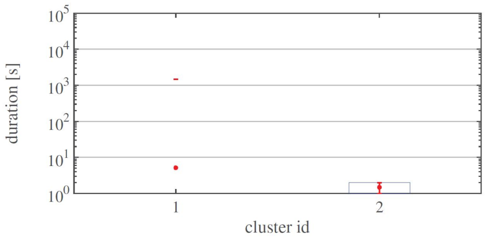
(a) Rate



(b) MCS



(c) RBS



(d) Duration

Figure 5.13: Study of dataset related to the campus area, as a whole.

Comments for the residential area: it is important to note that there is a strict relation between the number of sessions belonging to the cluster and the average data rate experienced by its traffic sessions. About 45% of sessions report an average data rate equal to 0.46 Mbps. Instead, only 3.26% of sessions register an average data rate equal to 5.74 Mbps. Intermediate average data rates refer to intermediate groups of sessions (i.e., 35.89% and 15.74% of sessions present an average data rate equal to 1.33 Mbps and 2.60 Mbps, respectively).

Similar behavior is observed for MCS indexes and the allocated RBs. Figure 5.12b shows that the selected average MCS index is lower than 4 for about 45% of sessions. Only 3.26% of sessions use an average MCS index close to 10. Moreover, 35.89% and 15.74% of sessions use an average MCS index approximately equal to 6 and 8, respectively. Only intermediate clusters register peaks of MCS up to nearly 26. Considering that LTE allows a maximum MCS index equal to 31, obtained findings clearly highlight that the channel quality experienced by mobile terminals during the monitoring is relatively scarce.

Interesting details related to the distribution of radio resources among mobile terminals are depicted in Figure 5.12c. About 45% of sessions, which have the average data rate equal to 0.46 Mbps, consume the lowest amount of physical resources. Considering that 100 RBs per TTI are available in 20 MHz bandwidth, an average number of RBs per TTI approximately equal to 23 means that sessions belonging to the first cluster occupy less than 1/4 of the overall amount of resources available within a cell. On the other hand, only 17 sessions consistently use a larger amount of resources per TTI, thus obtaining higher data rates. A quite different behavior emerges from the analysis of the average session duration. Sessions that register the average data rate equal to 5.74 Mbps remain active for about 40 s, which is the lowest amount of time among the four clusters. For other clusters, instead, the duration increases with the number of sessions belonging to the cluster. It is also important to note that the session duration always presents a very high variability: the actual duration of 75% of sessions in each cluster is lower than the one associated with the related centroid.

Comments for the campus area: the campus area presents a number of sessions extremely higher than the residential case, but the reported bandwidth requirements are extremely lower. As expected, there is a strict relation between the number of sessions per cluster and the average data rate. Nevertheless, almost all the sessions monitored in the campus area (i.e., 99.92%) fall within the same cluster and register a very low average data rate equal to 0.14 Mbps. Only 0.08% of sessions register an average data rate of 5.47 Mbps.

The study of MCS indexes provides a reverse relation, as shown in Fig. 5.13b. The former group of sessions experiences variable channel conditions, translating into the usage of all the admitted transmission settings. While the average MCS index is 13, the maximum value is equal to 31. The second group of sessions (4 out of 4946) registers worse channel conditions. In this case, the average and the maximum MCS indexes are about 7 and 10, respectively.

Fig. 5.13c confirms what observed for the residential area: the higher the average number of RBs used per TTI, the higher the achieved data rate. Reported results still show that 4942 sessions use about 1/4 of the bandwidth per TTI. On the contrary, only 4 sessions use a larger amount of resources per TTI (i.e., more than 52). As depicted in Fig. 5.13d, the campus area hosts sessions with very short durations. Apart from one exception (e.g., the graph reports one session duration equal to 1465 s), the former group of sessions registers an average session duration of 5 s. The duration of sessions belonging to the second cluster, instead, is lower than 2 s.

Study based on time-slots

cluster id	# of sessions	rate [Mbps]			MCS [#]			RBs [#]			duration [s]		
		min	max	avg	min	max	avg	min	max	avg	min	max	avg
morning (from 6:00 am to 12:59 pm, 58 sessions)													
1	17	0.08	0.37	0.22	1.59	3.56	2.68	11.85	24.29	17.90	12	69	19.88
2	23	0.46	0.97	0.65	2.96	7.07	4.59	19.11	37.75	26.14	12	2696	324.39
3	18	0.99	2.17	1.31	4.28	6.83	5.75	26.75	47.93	34.64	12	3304	367.11
afternoon (from 1:00 pm to 7:59 pm, 297 sessions)													
1	120	0.06	0.86	0.48	1.15	6.95	4.11	12.08	48.55	23.72	12	478	51.67
2	116	0.87	1.80	1.26	3.51	25.69	5.80	19.09	63.05	33.79	12	3716	158.72
3	53	1.84	3.39	2.39	4.27	21.28	6.99	27.78	61.86	40.99	12	9845	598.09
4	8	3.56	6.12	4.47	7.39	11.96	9.49	45.81	72.62	61.34	12	20	15.38
evening (from 8:00 pm to 12:59 am, 134 sessions)													
1	78	0.09	1.58	0.79	1.98	7.45	4.67	12.98	57.24	27.12	12	135	26.14
2	46	1.64	4.31	2.48	5.14	24.44	8.07	26.86	66.82	44.36	12	8600	571.87
3	10	4.80	8.07	6.26	7.19	11.74	9	42.10	70.52	57.83	12	41	17.60
night (from 1:00 am to 5:59 am, 31 sessions)													
1	27	0.02	0.71	0.30	0.75	6.72	3.38	10.45	44.89	23.19	12	14630	793.81
4	2	1.15	1.60	1.38	3.97	6.87	5.42	34.77	48.56	41.67	20	3416	1718
3	1	2.85	2.85	2.85	7.83	7.83	7.83	46.86	46.86	46.86	126	126	126
2	1	5.91	5.91	5.91	12.73	12.73	12.73	66.62	66.62	66.62	399	399	399

Figure 5.14: Study of the dataset related to the residential area, on time-slots basis.

cluster id	# of sessions	rate [Mbps]			MCS [#]			RBs [#]			duration [s]		
		min	max	avg	min	max	avg	min	max	avg	min	max	avg
morning (from 6:00 am to 12:59 pm, 1225 sessions)													
1	681	<0.01	0.16	0.06	0.79	29	12.54	5	47.33	24.18	<1	292	8.45
2	538	0.16	0.79	0.25	1.55	20.93	14.44	10.83	38.25	24.99	<1	11	<1
3	6	0.88	2.43	1.42	3.70	14.19	8.60	20.73	33.34	27.73	<1	908	151.33
afternoon (from 1:00 pm to 7:59 pm, 1134 sessions)													
1	425	<0.01	0.07	0.02	1	29	11.35	5	49	24.18	<1	1114	12.04
2	298	0.07	0.17	0.12	6.27	27	13.80	11.65	54.50	22.66	<1	<1	<1
3	278	0.17	0.27	0.21	2.65	21.12	14.32	11.66	40.08	24.60	<1	3	<1
4	99	0.27	0.44	0.32	2.34	20	14.38	13.62	36.95	25.43	<1	1465	23.10
5	32	0.45	1.10	0.57	3.44	17.65	13.72	18.73	31.36	24.86	<1	<1	<1
6	1	2.23	2.23	2.23	8.09	8.09	8.09	47.64	47.64	47.64	16	16	16
7	1	3.73	3.73	3.73	6.09	6.09	6.09	42.78	42.78	42.78	1	1	1
evening (from 8:00 pm to 12:59 am, 848 sessions)													
1	846	<0.01	0.82	0.13	0.58	31	13.09	4.50	52	23.41	<1	234	3.04
2	2	5.16	6.45	5.80	5	9.99	7.49	55.16	57.86	56.51	2	2	2
night (from 1:00 am to 5:59 am, 1738 sessions)													
1	627	<0.01	0.09	0.03	0.12	26.67	11.22	4.75	52	22.37	<1	695	12.39
2	690	0.09	0.22	0.16	0.72	23.11	14.08	8.77	45.25	23.62	<1	1	<1
3	363	0.22	0.42	0.28	1.90	21.29	14.66	11.77	36.63	25.31	<1	11	<1
4	57	0.42	0.98	0.56	2.92	16.82	14.07	16.14	31	25.01	<1	1080	19.14
5	1	6.56	6.57	6.56	8.36	8.36	8.36	53.74	53.74	53.74	1	1	1

Figure 5.15: Study of the dataset related to the campus area, on time-slots basis.

The analysis of mobile traffic on time-slots basis leads to a detailed characterization of sessions, with a consequent deep recognition of resource usage and QoS requirements that a mobile network has to address during different parts of the day. The outcomes of the proposed clustering methodology on time-slots basis, applied to both residential and campus areas, are summarized in Tables 5.14 and 5.15, respectively.

Comments for the residential area: The relation between the number of sessions belonging to the cluster and the average variable registered by related traffic sessions still exists. About 30% of morning sessions report an average data rate equal to 0.22 Mbps, while about 40% have an average data rate equal to 0.65 Mbps. During the afternoon, that is the time-slot with the highest number of residential sessions, the data rate starts growing. In fact, about 40% of afternoon sessions have an average data rate equal to 0.48 Mbps and the other 40% of sessions register an average data rate equal to 1.26 Mbps. The data rate still grows during the evening. The average data rate is 0.79 Mbps and 2.48 Mbps for about 60% and 35% of evening sessions, respectively.

Considering night sessions, whose number is limited because people tend to sleep, the average data rate goes down: about 87% of night sessions report an average value equal to 0.30 Mbps. The average MCS index is lower than 5 for about 70% of morning sessions. In particular, around 40% use an average MCS index close to 5 and around 30% even use an average MCS index approximately equal to 3. During the afternoon, average MCS indexes increase. In fact, about 40% of afternoon sessions have an average MCS index close to 4 and a further 40% register an average value close to 6. The MCS indexes still grow during the evening, as the data rate. The average MCS index is approximately 5 and 8 for about 60% and 35% of evening sessions, respectively. As regards night sessions, MCS indexes tend to reduce. In fact, about 87% of night sessions report an average value close to 3.

The distribution of radio resources follows a similar pattern. About 30% of morning sessions report an average number of RBs per TTI equal to 1/6 of the overall amount of resources available within a cell, while about 40% have an average value equal to 1/4. During the afternoon, about 40% of sessions use an average number of RBs per TTI close to 1/4 of bandwidth per TTI and a further 40% register an average number equal to 1/3. During the evening, the average amount of resources per TTI is more than 1/4 and about 1/2 of the overall bandwidth for about 60% and 35% of sessions, respectively. Then, bandwidth consumptions decrease during the night: about 87% of night sessions consume less than 1/4 of bandwidth per TTI.

The average duration, which varies greatly, has different behavior. Sessions generally register a short duration (i.e., 600 s), except for those available in the night time-slot. In fact, about 87% of night sessions last about 800 s. Moreover, around 6.5% have an average duration equal to 1718 s.

Comments for the campus area: the campus area reports a more balanced distribution of sessions among the wholeday slots. As expected, traffic classification on time-slots basis offers a better characterization of sessions. For example, up to 7 clusters are identified for the afternoon time-slots, against the only two clusters reported for the analysis of the dataset as a whole. Regarding the data rate in the campus area, more than 55% of morning sessions report an average data rate equal to 0.06

Mbps, while about 40% have an average data rate equal to 0.25 Mbps. During the afternoon, the data rate tends to decrease. In fact, about 40% of afternoon sessions report an average data rate equal to 0.02 Mbps and about 26% register an average data rate equal to 0.12 Mbps. The data rate is still low during the evening. In fact, the average value of 0.13 Mbps is measured for more than 99% of sessions. During the night, that is the time-slot with the highest number of sessions, the average data rate tends to increase. From Table 5.15, it is 0.03 and 0.16 Mbps for about 36% and 40% of night sessions, respectively.

The average MCS index is similar among the time-slots. In particular, about 55% of morning sessions have an average value close to 13. About 40% of afternoon sessions register an average MCS index close to 11, while about 26% and 25% use an average MCS index approximately equal to 14 and more than 14, respectively. During the evening, the average MCS index is approximately 13 for 99.76% of sessions. Lastly, it is close to 11 and 14 for about 36% and 40% of night sessions, respectively.

Also the allocated RBs per TTI have similar behavior. In particular, they slightly increase and decrease during the morning and the afternoon and during the evening and the night, respectively. About 56% of morning sessions and 38% of afternoon sessions report an average number of RBs per TTI close to 1/4 of the overall amount of resources available within a cell. Instead, the average amount of resources per TTI is more than 1/4 of the overall bandwidth for 99.76% of evening sessions and about 76% of night ones.

The average duration is extremely low during all the considered time-slots. In particular, about 56% of morning sessions last about 9 s. Furthermore, about 38% of afternoon sessions last longer than 10 s (i.e., about 12 s), while more than 50% (the clusters 2 and 3 in the afternoon) last less than 1 s. The average duration is approximately equal to 3 s for about 99% of evening sessions. As the last report, about 36% of night sessions last longer than 10 s (i.e., about 12 s), while around 60% last less than 1 s.

5.2.3 Considerations

The proposed study clearly shows that the analysis of mobile traffic on time-slots basis gives a deep insight into radio resource utilization dynamics. Obtained results report a clear heterogeneity among traffic sessions, whose clustering offers key instruments for the optimal management of the radio resources for mobile operators.

As far as the residential area is concerned, a high number of sessions are measured for the afternoon time-slot and peaks of bandwidth requirements are registered in both afternoon and evening time-slots. By observing data related to the night time-slot, it is possible to understand that a residential area significantly reduces its traffic load when people usually go to sleep. Nonetheless, differently from daily time-slots, the few sessions active during the night present very high durations. Regarding the campus area, sessions use a higher MCS index than residential sessions, but a very low rate: analyzed campus sessions do not transmit a lot of data, even if the quality of channel could be good, because the traffic load is not significant.

By knowing the radio resource utilization patterns of mobile traffic, it will be possible to conceive novel methodologies that aim at optimizing mobile networks. Interesting research activities to address in the future may include:

- Advanced QoE/QoS management through dynamic radio resource scheduling algorithms exploiting the deep properties expected for mobile flows at the radio level;
- Dynamic and fine-grained management of slices and virtual functionalities offered through the radio access networks in upcoming 5G architectures;
- Optimal energy savings mechanisms (e.g. sleep mode of base stations and discontinuous reception in mobile terminals) and opportunistic handover management procedures that leverage the predicted behavior of classified traffic flows;
- Planning for new base station deployments in geographical regions where higher traffic load is expected.

5.3 Mobile Traffic Prediction from Raw Data Using LSTM Networks

Predictive analysis on mobile network traffic is becoming of fundamental importance for the next generation cellular network. Proactively knowing the user demands, allows the system for an optimal resource allocation. In this Section, we study the mobile traffic of an LTE base station and we design a system for the traffic prediction using Recurrent Neural Networks. The mobile traffic information is gathered from the LTE PDCCH using the passive tool presented in[16].

The design of the prediction system includes Long Short Term Memory units. With respect to a Multilayer Perceptron Network, or other artificial neurons structures, recurrent networks are advantageous for problems with sequential data (e.g. language modeling) [47]. In our case, we state the problem as a supervised multivariate prediction of the mobile traffic, where the objective is to minimize the prediction error given the information extracted from the PDCCH. We evaluate the one-step prediction and the long-term prediction errors of the proposed methodology, considering different numbers for the duration of the observed values, which determines the memory length of the LSTM network and how much information must be stored for a precise traffic prediction.

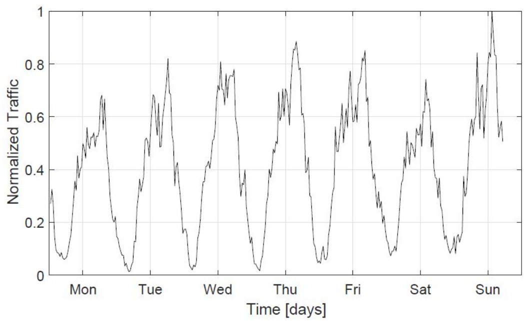
5.3.1 The LTE Scheduling Dataset

The collected dataset consists of one-month of scheduling information that we gathered by monitoring different eNodeBs located in the city of Barcelona, Spain. Let $\mathcal{D} = \{D_{c_1}, D_{c_2}, ..\}$ be the dataset, where D_{c_k} is the set of measurements for the monitored cell k. Given a D_{c_k}, for each connected user at the time t, temporary identified by a RNTI r, we decode the DCI message containing the resource blocks, the transport block size and other scheduling information for the uplink and the downlink directions. We store this information in a measure $\mathbf{s}_r(t)$, where t corresponds univocally to an LTE subframe number, or TTI, which is 1 ms long.

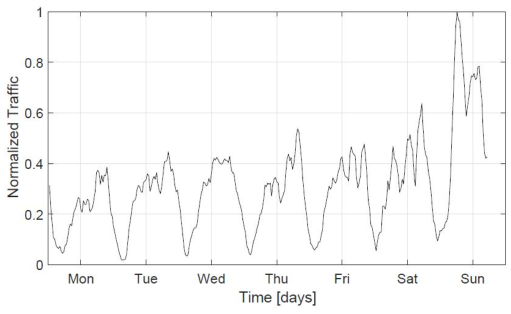
We calculate the aggregate cell traffic measurements for a given timeslot T, which is the sum of the traffic generated by all the RNTIs connected during the timeslot T, $R(T)$

$$\mathbf{S}(T) = \sum_{r(t) \in R(T)} \sum_{t \in T} \mathbf{s}_r(t) \tag{5.2}$$

Thus, $\mathbf{S}(T)$ is the vector that contains the number of resource blocks allocated in the uplink and in the downlink directions, the number of the messages sent in both the directions and the sum of the total transport block sizes for a given timeslot T. Moreover, for each timeslot T, we include the number of the attached users to the eNodeB in the timeslot T. In our case we consider T, as the number of TTIs for which we aggregate the traffic.



(a) eNodeB 1



(b) eNodeB 2

Figure 5.16: Normalized weekly traffic signature of two monitored LTE eNodeBs.

5.3.2 A Long-Short Term Memory Network

Recurrent neural networks are a generalization of feedforward neural networks, that have been devised for handling temporal and predictive problems. LSTM are a particular kind of RNN, that have been introduced in [101]. They have been explicitly designed to avoid the long-term dependency issue, which is the cause of the vanishing-gradient problem in normal RNNs [102].

The capability of learning long-term dependencies is due to the structure of the LSTM units, which incorporates gates that regulate the learning process. In a standard LSTM unit, the basic operations are accomplished by the input gate $\mathbf{i}_t$, the forget gate $\mathbf{f}_t$ and the output gate $\mathbf{o}_t$. Moreover, the cell state $\mathbf{c}_t$ represents the memory of the unit and it is updated with the information to be kept (or to be forgotten), provided by the input gate (or forget gate).

The LSTM unit combines the output of the previous unit $\mathbf{h}_{t-1}$ with the current input $\mathbf{x}_t$ using the input, the output and the forget gates to update the memory of the cell. The variables $\mathbf{i}_t$ and $\mathbf{f}_t$ represent respectively the information that need to be kept or to be forgotten from the past and the current input. The cell state $\mathbf{c}_t$ is updated by summing the previous cell state $\mathbf{c}_{t-1}$ and the candidate cell state $\tilde{\mathbf{c}}_t$, weighted respectively with $\mathbf{f}_t$ and $\mathbf{i}_t$. Finally, we obtain the output $\mathbf{h}_t$ applying the tanh function to $\mathbf{c}_t$ and multiplying it by $\mathbf{o}_t$. Then, the current output $\mathbf{h}_t$ is passed to next unit and combined with the input at the next time index $t + 1$.

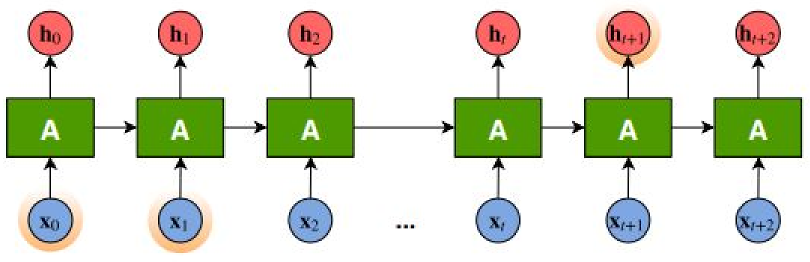


Figure 5.17: Single-layer LSTM network.

Multiple LSTM units are concatenated to form one layer of the LSTM network. Each unit computes the operations on one time index and transfer the output to the next LSTM unit. The number of concatenated cells indicates the number of observations of the data that are considered before making the prediction. In our case, the input $\mathbf{x}_t$ is the eNodeB traffic vector $\mathbf{S}(T)$, and the number of observations is the number of selected timeslots T.

The proposed architecture for the mobile traffic prediction is depicted in 5.18. In our design, we consider multiple layers of basic LSTM units to form a stacked LSTM network. The intuition is that the deep LSTM network is able to learn the temporal dependencies of the aggregate mobile traffic: the LSTM unit of each layer extract a fixed number of features which are passed to the next layer. The depth of the network (e.g. the number of layers) is to increment the accuracy of the prediction, which is done by the last fully connected layer.

For the one-step prediction we use a many-to-one architecture, which means that the network observes the mobile traffic for a fixed number of timeslots until T and, then try to predict the traffic in the next time slot $T + 1$. In the multi-step prediction, we delay the prediction for a chosen number of timesteps, similarly to what is

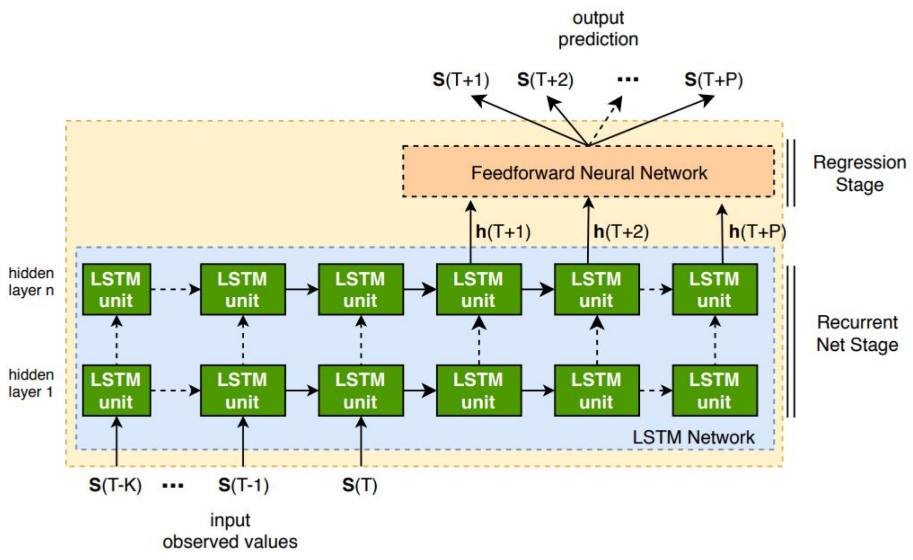


Figure 5.18: Proposed architecture for the mobile traffic prediction.

done for language modeling problems when they try to predict a sequence of words. Last, the output of the LSTM network is passed to a fully connected neural network, which performs the actual prediction.

5.3.3 LSTM Network Performance

Evaluation Setup We use the set of mobile traffic data from two different eNodeBs, that we collected during one month, to evaluate the performance of the proposed architecture. For each eNodeB, we calculate the aggregate cell traffic, as described in the previous section. We choose the Normalized Root Mean Square Error (NRMSE) as the metric to measure the accuracy of the prediction algorithm, which is given as

$$\text{NRMSE} = \frac{1}{\bar{x}} \sqrt{\frac{\sum_{t=1}^{N} (\tilde{x}_t - x_t)^2}{N}} \tag{5.3}$$

where N is the total number of points, $\tilde{x}_t$ and x_t are the predicted value and its correspondent observation at the time t and $\bar{x}_t$ is their mean. This same metric is used to compare the accuracy of the proposed architecture with the one obtained using other predictive algorithms.

The implementation of the mobile traffic prediction algorithm is done in Python, using Keras and Tensorflow, as backend. The chosen hyperparameters are reported in Table 1. The number of hidden layers is fixed to 5: this is one of the hyperparameters that need to be selected and can affect the tradeoff between the prediction accuracy and the time needed to train the network. A higher number of layers may increase the precision of the prediction, but we want to focus on the the relationship between the number of past observed values and the precision of the multi-step prediction, which determines the quantity of information needed to be memorized and utilized by the network. For the same reason, we fix the number of epochs to 100. Three weeks of data are used to train and to validate the architecture. Next results are related to the last week. We use the Adam optimization [116], to update the network weights iteratively based on the training data.

Table 5.1: Training Hyperparameters for the LSTM Stacked Network

Initial Learning Rate	0.001
Num. of Epochs	100
LSTM Hidden States	64
LSTM Hidden Layers	5
Feedforward Hidden Layers	1
Optimization Algorithm	Adam
Loss Function	MAE

Results Analysis Next, we present the results of multi-step prediction, that is when the output is delayed for a fixed number of timeslots and the prediction is performed for later time instants. We show how the accuracy decreases when we try to predict the traffic data in future timesteps. Furthermore, we analyze the effect of the number of observations that the LSTM network can see, and the duration of the timeslots T: these are design parameters that need to be estimated, since they determine the memory length of the LSTM network and how much traffic information is needed to be stored for an accurate prediction.

In 5.19, we show the results of the mobile traffic prediction for two cells: since they are located into two different areas, the monitored eNodeBs present two distinct traffic profiles in terms of profile and traffic magnitude. We can see that the prediction is precise for the whole week, despite the oscillating behaviour of the traffic. In this case, the prediction is one-step ahead, that means that we use a fixed number of past values $(K = 10)$ to predict the traffic for the next timeslot.

In 5.20a and in 5.20b we evaluate the prediction error with respect to past observed values. It is relevant also to consider different values for the timeslot duration of T, which affects the calculation of the aggregated traffic $\mathbf{S}(T)$ from the raw LTE traces. Figures are related to the first eNodeB (results are comparable for eNodeB 2). In Fig. 7, we see that the NRMSE is larger for a higher duration of T and, as expected, the error decreases with a larger number of observations. To emphasize the effect of the number of past observations, we plot the increasing accuracy (with respect to observing only one past value) for different values of T. We can observe

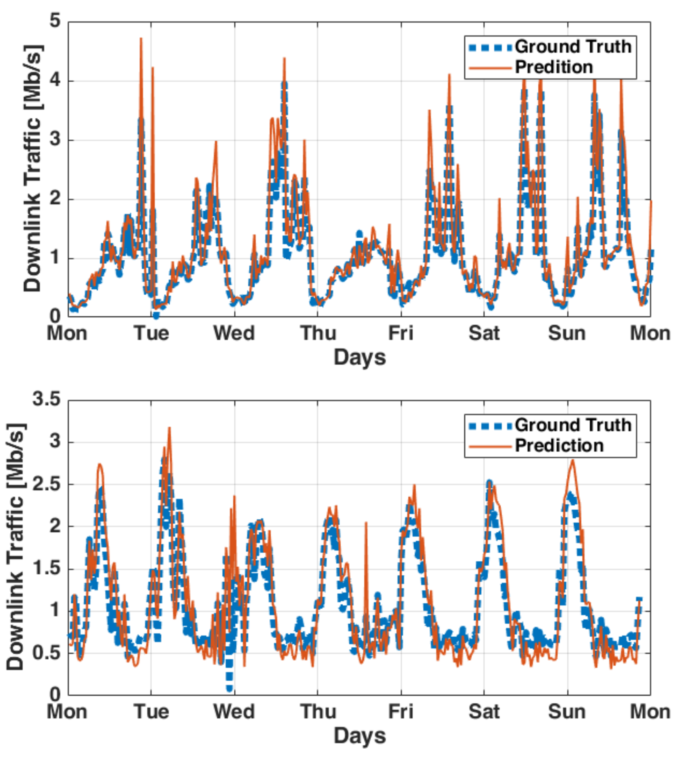
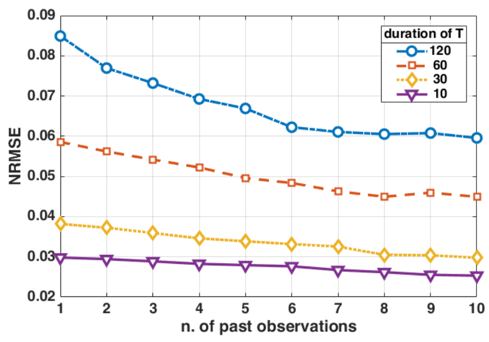
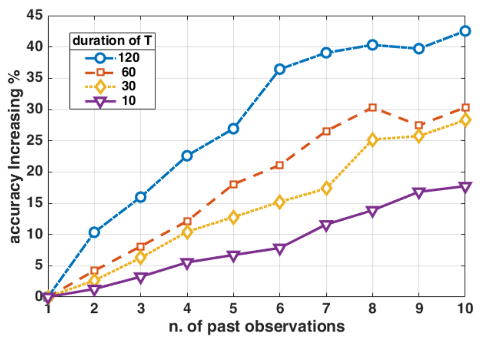


Figure 5.19: Prediction of the weekly mobile traffic for two different eNodeBs.

that the major increase in percentage is given for larger values of T. For 10 past observed timesteps the accuracy can increase more than 40%.



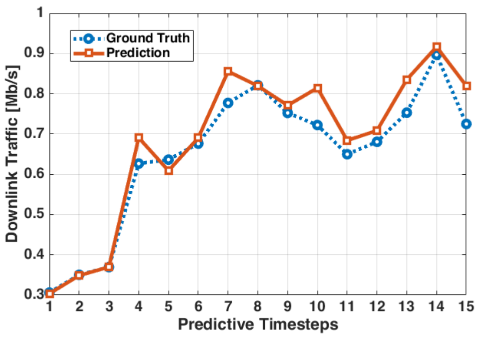
(a) NRMSE.



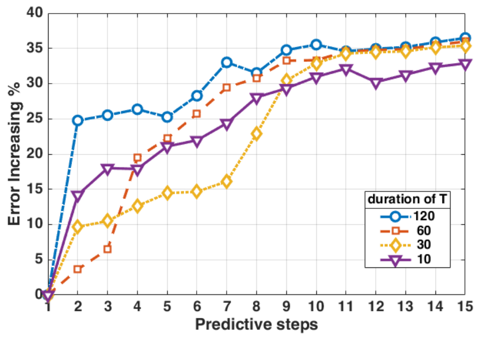
(b) Percentage of accuracy gaining.

Figure 5.20: Error and accuracy gaining vs number of past observed values.

In 5.21a we show the prediction for 15 timesteps ahead. We fix $K = 10$ and $T = 10$ TTIs. At first, the prediction almost corresponds to the measured values, while after some steps the prediction error is more dominant. In 5.21b, we plot the increasing error with respect to future prediction steps. As expected, longer prediction causes an increment in the error of the algorithm. The error increases with different trends and it is around 40% when we predict for 15 steps ahead. This is similar to what happens in the problem of language synthesis for the prediction of long sentences: for further words prediction, the number of candidate words is larger, therefore there is more uncertainty in the correct choice. Conversely to the number of past observations, changing the duration of the timeslot T, does not give useful insight on the prediction error: for longer periods, the variability of the mobile traffic is larger, leading to an oscillating and randomic error for future predictions.



(a) Predicted vs. ground-truth for 15 steps ahead ($T = 10$).



(b) Error increasing vs number of predicting timesteps.

Figure 5.21: Lookup for the predicted mobile traffic and error increasing for 15 steps ahead.

Comparison with ARIMA Finally, we compare the proposed architecture with two time-series prediction methods: an ARIMA model is a well-established technique for the time-series analysis, and it is defined by 3 parameters (p, d, q) that determine the auto-regression, the differentiation and the moving average, respectively. Here, we use a $(10, 1, 5)$ model. Also, note that we use only one variable for the prediction (i.e. the aggregate traffic), instead of multiple information obtained by the raw data. The other traffic prediction is obtained using a deep FeedForward Neural Network (FFNN), where we replace the LSTM neural network with a network of fully connected neurons. For a fair comparison, we use the same number of hidden layers. 5.22 shows the traffic prediction on the same time window using these two techniques. The accuracy using the ARIMA model is lower, since the prediction tends to be closer to the average value of the traffic. On the other hand, the FFNN is able to follow the periodic trend and the traffic oscillations, but it still lack of a high precision. Also, we compare the average error for the three prediction methods on the two traffic profiles: as expected, thanks to the LSTM properties, the proposed algorithm captures the temporal characteristics of the mobile traffic, and it provides superior accuracy with respect to the FeedForward Neural Network or to the classic ARIMA model.

Conclusions We study the effectiveness of recurrent neural networks applied to the prediction of the mobile traffic. The choice of using LSTM network is imposed by the dataset characteristics, since we use multivariate traffic information that derive

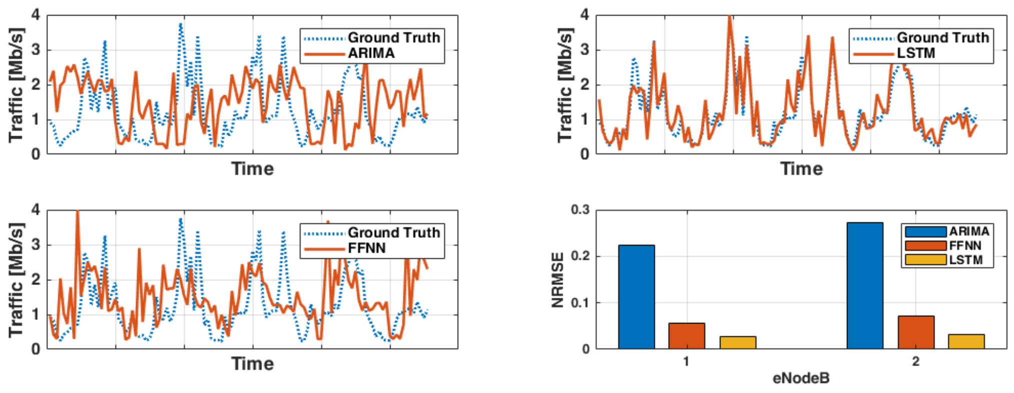


Figure 5.22: Traffic prediction obtained with different model and errors.

directly from the DCI of the LTE control channel. The LSTM units succeed in capturing the temporal correlation of the traffic even for distant timeslots. Applying the prediction of the traffic using raw aggregate data from the physical channel, is fundamental in time-critical applications and avoids the need for additional resources to process the traffic data.

Chapter 6

Mobile Traffic Anomaly Detection in Urban Environments

In this Chapter, we evaluate the possibility of exploiting the pervasiveness of mobile newtorks and use them as sensing platforms to detect potential urban anomalies. First, in Section 6.1, we use the acquired data from the Camp Nou area, to implement a supervised algorithm to detect the users' behaviours when a football match is scheduled. Next, in Section 6.2, we extend the detection of anomalies to unknown events, using a semi-supervised approach, in which only normal traffic samples are used to train the algorithm. The results reported in this Chapter are presented in the following papers:

- Trinh, H. D., Giupponi, L., & Dini, P. (2019, June). **Urban anomaly detection by processing mobile traffic traces with LSTM neural networks**. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON) (pp. 1-8). IEEE.* [22]

- Trinh, H. D., Zeydan, E., Giupponi, L., & Dini, P. (2019). **Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach**. *IEEE Access, 7, 152187-152201.* [23]

6.1 Detecting Anomalies through Supervised Approach: the Case of Barcelona Camp Nou

Nowadays most of the population of the planet lives in towns with a higher and higher concentration of people in the metropolitan areas. This trend brings evident economic benefits to citizens, but, several issues for their wellness, e.g., congested mobility and transportation, air quality and pollution, social segregation, to name a few, have been raising. Therefore, such demographic changes require cities to implement smart strategies for a more sustainable development and management of metropolitan areas.

In this context, the automatic detection of urban anomalies, like unexpected crowd gathering, is of upmost importance for government and public administration [48]. However, urban anomalies often exhibit complicated forms, and monitoring heterogeneous sources like traffic flows or public transportation usage, requires complex sensing systems. Generally, the collection of such information can be achieved with a remote sensing platform, composed of a distributed network of sensors and cameras [117], or, alternatively using crowd-sourcing methods [118]. However, enabling such complex platforms requires the direct human intervention and it can be expensive due to the installation and to the maintenance of the hardware needed to monitor and report the public status in different parts of the city.

6.1.1 Mobile Network as a Sensing Platform

A viable opportunity to effectively complement the already available monitoring systems, is to exploit the extreme pervasiveness of the mobile networks: using the mobile network as a sensing platform, eliminates the need for additional expensive hardware and it is valuable in the long-term because of 5G Ultra-Dense Networks (UDN), which, in the upcoming years, will boost the ubiquity of the mobile networks [119].

In this work, we demonstrate how to perform Anomaly Detection (AD) using mobile network data: to this end, we leverage a Multi-access Edge Computing (MEC) architecture, which enables the mobile data processing directly from the radio access, and the detection of anomalies occurring in an area covered by one base station. The mobile data is collected from the LTE Control channel and it is provided to a MEC server, which promptly processes the information close to the edge of the network, i.e. at the radio-access, avoiding high latencies. MEC provides service , through the virtualization of network functions that formerly existed in the Evolved Packet Core (EPC): the movement of these functionalities close to the base station enables a boost in the performance and of the Quality of the Services (QoS). The benefits are due to a network deployment based on the virtualization of the network functions, which formerly existed in the Evolved Packet Core (EPC), and moved to the network edge close to the base stations.

The approach we adopt is the following: we collect the mobile network data by passively sniffing the unencrypted LTE PDCCH from base stations in a certain area. We proceed by identifying a known event (e.g. a football match) that is expected to generate a large concentration of people in a certain urban zone. We collect measurements from the target zone during different days for a sufficiently long period, and finally we design and use an anomaly detection tool that is able to identify the anomalous behaviour, during the targeted event. The identification of such unexpected events is beneficial in a wide range of contexts, for example, for public safety purposes, for the optimization of urban planning, and for network management optimization, to handle e.g. network congestion issues that may affect the Quality of experience (QoE) of the users, especially if they are demanding real-time services.

6.1.2 Objectives

We design an anomaly detection (AD) system based on RNNs, which are the state-of-the-art learning techniques to cope with sequential input data, showing outstanding performance, for example, in the area of Natural Language Processing (NLP) [120]. We adopt Long Short-Term Memory (LSTM) neural networks, which are capable of learning long-term dependencies from the input time series, while solving the vanishing-gradient problem that affects standard RNNs.

The presented analysis shows that our proposed algorithm for AD achieves an F-score of 1 on the considered dataset and also provides a comparison with other classes of algorithms. The methodology and the achieved results are novel in the context of urban anomaly detection. In summary, the original contributions are the following:

- *Mobile Network as a Sensing Platform:* we propose to exploit the pervasiveness of the existent network to monitor locally the presence of people and to detect potential anomalies; this method reduces the need of installation and maintenance of additional expensive hardware;
- *Anomaly Detection with LSTM Neural Networks and Comparison*: we design an algorithm based on LSTM neural networks and we tune the training parameters to obtain a maximum F-score of 1. The algorithm is intended to work in real-time, based on the LTE control data provided to the MEC server. A comparison with other state-of-the-art algorithms demonstrates the advantage of our supervised approach.

6.1.3 Scenario

We consider a scenario like the one depicted in Fig.6.1, where a MEC server is deployed and co-located with a multi-access RAN, e.g. LTE base stations (eNodeB). The MEC server coordinates several virtual machines (VMs), which share the computational efforts to support the traffic load from a limited number of eNodeBs and it is provided with the LTE network data. To supply this information to the MEC, one solution is to create a link to share the internal base station data: however, to get access to the eNodeB information, it is required the direct intervention of the mobile network operator.

Alternatively, the solution we adopt consists of listening to the LTE Control Channel information and it can be feasibly performed by external individuals: using the

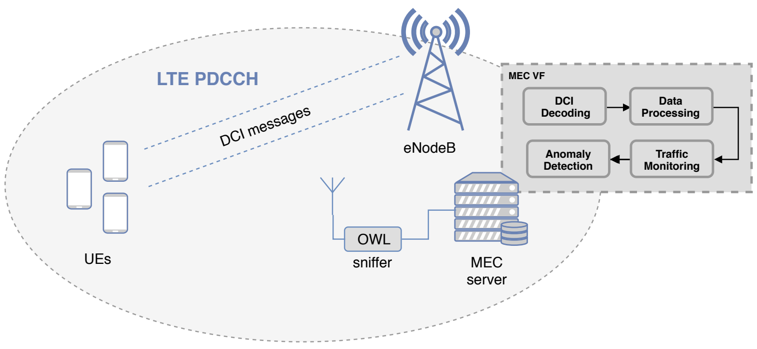


Figure 6.1: Scenario.

unencrypted data sent over the LTE PDCCH we can obtain the full scheduling information about the radio resource usage for that particular base station. More precisely, it is possible to serve MEC with DCI messages, from which we can derive the resource blocks and the modulation and coding index assigned to the users together with an user identifier. This approach presents two main advantages:

1. the collected data does not present any privacy/security issue, since it relies on the LTE security protocols, and therefore no additional procedure to preserve user's anonymity is required;

2. the passive listening over-the-air does not need additional expensive hardware to provide the information to the MEC server and the DCI decoding can be performed directly using open-source software.

For these reasons, the proposed approach exploits the existent infrastructure and is feasible in terms of costs and effectiveness.

6.1.4 Dataset

The measurement campaign took place in the city of Barcelona for one month. We monitored an eNodeB located nearby the popular Camp Nou football stadium: Camp Nou is the largest European football stadium and allows up to almost 100 thousand attendance per event. The stadium is located in a urban residential area of Barcelona, which is characterized by a high population density. The choice of the eNodeB to be monitored is made based on the high variability of the traffic during sports and leisure events, which are hosted periodically into the stadium.

In this work, we are interested in studying the total traffic exchanged between the eNodeB and all the connected users. Thus, we need to aggregate the eNodeB traffic: let $\mathcal{T}$ be the total measurements period; for every second $t \in \mathcal{T}$, we define $x(t)$ as the vector that contains the following information

1. RNTI: the total number of assigned C-RNTI;

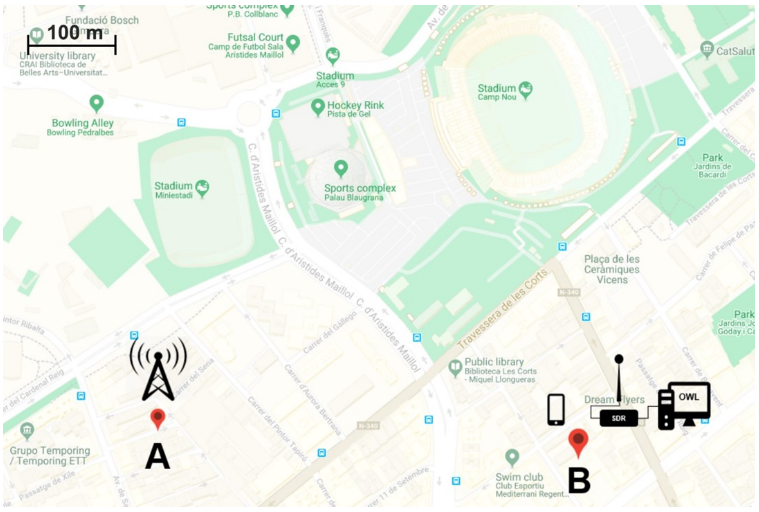


Figure 6.2: Map of Barcelona area where the measurement campaign took place for the creation of the input dataset. The eNodeB location is denoted by A, whereas the data collection system is marked as B.

2. TBS_{down}: the total number of transport block size assigned in the downlink direction;
3. TBS_{up}: the total number of transport block size assigned in the uplink direction;
4. RB_{down}: the total number of resource blocks allocated in the downlink direction;
5. RB_{up}: the total number of resource blocks allocated in the uplink direction;

We indicate with D the number of metrics we consider in $\boldsymbol{x}(t)$. Therefore, the sequence $\boldsymbol{x}(t)$ is a multi-variate time-series, which includes the metrics described above, which are extracted directly from the decoded DCI messages and aggregated over all the assigned C-RNTI.

6.1.5 Temporal Traffic Analysis

The collected dataset allows for a localized characterization of the mobile traffic, which is exchanged in the area of the monitored eNodeBs.

In Fig. 6.3, we observe the mobile traffic for several days. The plot includes one matchday, during which a football game takes place: it is possible to identify a regular daily pattern, but, also, traffic anomalies, which deviate from the normal behavior. The plots in Fig. 6.3 show the number of assigned C-RNTIs, the transport block sizes and the number of allocated resource blocks, averaged over a 30-minutes window. We recognize the match-day, due to the presence of prominent peaks. As observable, the number of C-RNTIs seems to represent a good indicator for measuring and detecting the traffic variations, since it is assigned by the eNodeB to temporarily identify the different UEs.

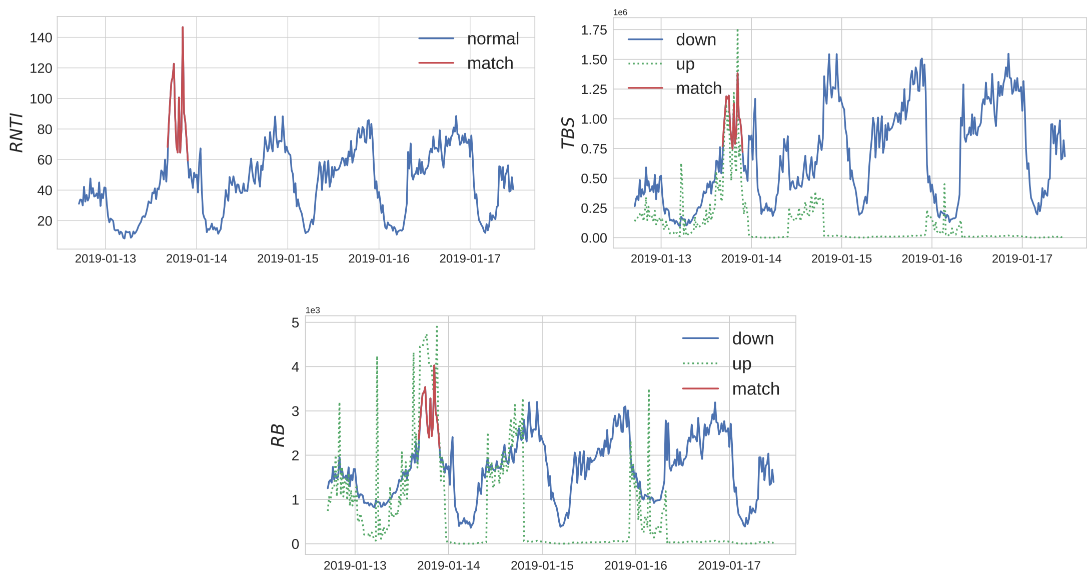


Figure 6.3: Traffic profiles including one match day.

Based on the previous observations, we establish that the traffic conditions that the network experiences can be categorized at least into two states, which are identified as *normal behaviour*, typical when no football match is scheduled, and as *anomalous behaviour*, when the traffic deviates from the expected behaviour at a given time of the day.

6.1.6 Football Match Insights

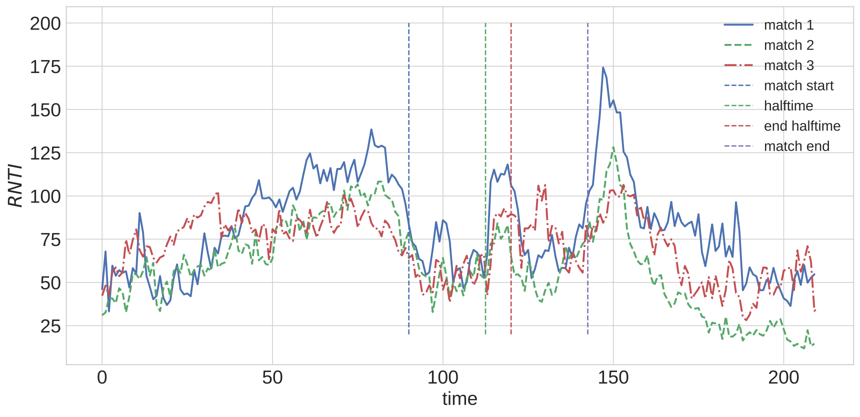


Figure 6.4: Matchdays traffic profiles.

One of the advantage of exploiting the LTE PDCCH information is the high resolution of the dataset, i.e. 1 ms, which allows for a detailed characterization of the mobile traffic. In the previous section, we show the daily traffic profile. Further meaningful insights can be acknowledged from the collected data.

In Fig. 6.4, we plot the traffic profiles of 3 different matches. Since the football

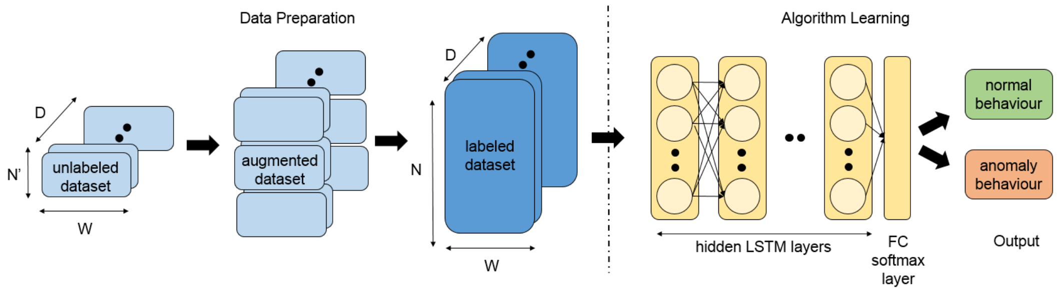


Figure 6.5: LSTM-AD Framework.

games took place at a different time of the day, we overlap the mobile traffic time-series in correspondence to the match kick-off. Also in this case, we plot the number of assigned C-RNTIs. We select a time-window that starts 3 hours before the match and terminates 2 hours after the match ends. The following observations can be made:

- *repetitive patterns*: we found that the same traffic pattern repeats during match-days, independently from the starting hour of the match;
- *traffic increase before the game*: this can be observed since we considered a large time resolution that start few hours before the game. The traffic increases because of the number of attendees that enter the areas nearby the stadium before the game;
- *traffic drop when the match starts*: it is that when the football match starts, the attendees start watching the game, and, therefore, they do not use their terminals;
- *half-time peaks*: as soon as the half time takes place, the attendance are prone to use their mobile phones; the mobile traffic drops as the half-time finishes, similarly to when the match starts, but more abruptly.
- *end-game peak*: after the match end, the traffic experiences the most prominent peak.

In the next section, we provide the description of an algorithm, whose objective is to detect the anomalous behaviour and make the network aware of the rapid traffic fluctuations.

6.1.7 AD-Framework

The problem of discriminating the anomalous states from a normal state of the network traffic conditions is classified as an anomaly (or outlier) detection problem. In this work, we design an anomaly detection (AD) system based on stacked Long Short-Term Memory (LSTM) neural networks, which are capable of tracking long-term dependencies from multi-variate time-series, while solving the vanishing-gradient problem that affects standard RNNs. The intuition is that a stacked LSTM network is able to extract the temporal dependencies of the mobile traffic patterns and learn to discriminate the anomalies from the normal pattern.

Since we know the match times and therefore, we know when the related anomaly occurs, the approach that we use in this work is supervised. Thus, the AD problem is addressed as a binary classification problem, where the designed algorithm is in charge of classifying the network traffic sequences into two classes: *normal* or *anomalous* behaviour. In general, AD supervised algorithms lead to more accurate results with respect to other techniques [121].

The whole framework to solve the AD problem is depicted in Fig. 6.5; it takes as input the data collected from LTE PDCCH and it consists essentially of 2 parts: *Data Preparation* and *Algorithm Learning*. The implementation details of both parts are discussed in the next sections.

6.1.8 Data Preparation

The dataset $\boldsymbol{x}(t)$, needs to be preprocessed and labeled before being input to the AD algorithm. Next we illustrate the procedure we adopt, consisting into four steps:

Data Resampling and Normalization

The sequence $\boldsymbol{x}(t)$ is resampled using a value t_s and standardized by removing the mean and by scaling to unit variance. This operation filters the input sequence and normalizes the original curve, henceforth it helps to identify rapidly the anomalies also by visual inspection (in Fig. 6.3 $t_s = 30$ minutes). Hereafter, to keep simple the notation, we use $\boldsymbol{x}(t)$ to also indicate the resampled sequence.

Data Windowing

The sequence $\boldsymbol{x}(t)$ is split and grouped using a fixed-length window W. The window is moved each time by one-step. The value of W defines the number of time-lags that the LSTM architecture processes to classify the input as anomalous or normal. This also determines the input length to the first LSTM layer.

The multi-variate sequence can be expressed as $[\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)]$, being T the cardinality of $\mathcal{T}$, i.e. $T = |\mathcal{T}|$. After the split, we have $N' = T - W + 1$ sequences $\mathbf{x}(n'), n' \in [1, N']$

$$\mathbf{x}(1) = [\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(W)]$$

$$\mathbf{x}(2) = [\boldsymbol{x}(2), \ldots, \boldsymbol{x}(W+1)]$$

$$\mathbf{x}(N') = \left[\boldsymbol{x}(N'), \ldots, \boldsymbol{x}(T)\right]$$

A sequence $\mathbf{x}$ has length W and each of its element is D-dimensional, with $D \in [1,5]$ the number of considered metrics described in Section 6.1.4. Hereafter, we refer to the sequence $\mathbf{x}$ as *samples*. Then, we can define $\mathbf{X}'$ the three-dimensional matrix which contains N' sequences $\mathbf{x}$. The matrix $\mathbf{X}'$ has dimension $N' \times W \times D$.

Data Augmentation

One of the most common problems in supervised classification is the lack of sufficient labeled data for which the algorithm is able to learn and distinguish the different classes [122]. This becomes more serious for anomaly detection, where, by definition, the anomalous behaviour appears very infrequently, creating a very unbalanced ratio between the two classes. Moreover, in order to validate the algorithm performance, we need to split the dataset into training and validation sets, which reduces the number of anomalous samples from which the algorithm can learn (or validate), making it difficult to obtain meaningful results in a statistical sense.

To overcome this issue, as done in [123], we can augment our dataset by oversampling the data by a factor F. The objective is to have enough samples in both the training and validation sets. This simple but effective method does not change the distribution of the anomalous samples (less than 8% in our case) in the dataset, but allows for statistical measure of the performance metrics (described in Section 6.1.10), which is required to evaluate the proposed algorithm.

We obtain $\mathbf{X}$ as the repetition of $\mathbf{X}'$: this operation can be seen as stacking $\mathbf{X}'$ F-times along the first dimension. The new matrix dimensions are $N \times W \times D$, with $N = N' \cdot F$. We choose $F = 3$ to have a sufficient number of anomalies (> 10) in the validation set. Similarly to $\mathbf{x}(n')$, we refer to the samples of $\mathbf{X}$ as $\mathbf{x}(n)$, $n \in [1, N]$.

Labeling

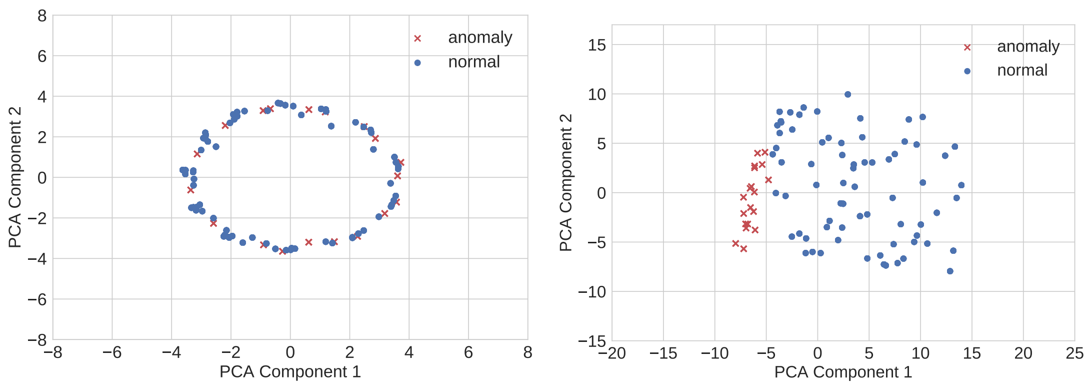
The dataset is labeled under the assumptions that we know when an anomaly occurs. In our approach, we define as an anomalous behaviour those traffic patterns that occur during a football match. As seen in Section 6.1.5, the network traffic deviates from his normal behaviour when there is a football game. This approach does not involve any threshold set or additional manual intervention. Moreover, this represents the best-effort approach in this case, since we strictly define what we consider an anomaly.

For each sequence $\mathbf{x}$ of $\mathbf{X}$, we assign a label of 1 if any of its elements is measured during the period of a football match and 0 in the opposite case. Note that the traffic patterns associated to the football match can occur at any time-step of a given $\mathbf{x}$, making the classification problem more complex.

6.1.9 LSTM-AD Architecture

The proposed architecture for urban anomaly detection is shown in the last part of Fig 6.5 and is based on Long Short-Term Memory (LSTM) neural networks. The capability of learning long-term dependencies is due to the structure of the basic LSTM cells (or units), inclusive of gates that regulate the learning process (see Fig. 7.16).

Multiple LSTM cells are concatenated to form one layer of the LSTM network. Each cell computes the operations on one time index and transfers the output to the next cell. The number of concatenated cells indicates the number of observations of the data, which in our case, corresponds to the window length W.



(a) Raw data. **(b)** Hidden features before softmax.

Figure 6.6: Scatter plots obtained with PCA of the raw input data vs the hidden features extracted by the last LSTM layer before softmax.

Table 6.1: LSTM-AD performance for different configuration of the input data with $W = 8$.

D	N	Best Input Combination	Prec	Rec	F-Score	W_{min}
1	5	[RNTI]	0.96	0.95	0.95	14
2	10	[RNTI, TBS_{down}]	0.95	0.96	0.96	11
3	**10**	**[RNTI, TBS_{down}, TBS_{up}]**	**1.00**	**1.00**	**1.00**	**8**
4	5	[RNTI, TBS_{down}, TBS_{up}, RBS_{down}]	1.00	1.00	1.00	8
5	1	[RNTI, TBS_{down}, TBS_{up}, RBS_{down}, RBS_{up}]	1.00	1.00	1.00	8

In our design, we consider a stacked architecture combining $L = 3$ LSTM hidden layers and a final Fully Connected (FC) layer. The L LSTM layers are composed of $H = 200$ LSTM units, while the last FC output layer is formed by 2 hidden neurons to perform the binary classification. In each LSTM layer, the hyperbolic tangent (tanh) activation function is adopted to process the output to be passed to the subsequent layer. Differently, in the last FC layer, a *softmax activation function* produces the final output, which corresponds to the probabilities of belonging to the anomaly class or to the normal class. Finally, the classification is performed by picking the class with the highest likelihood probability. The algorithm is trained using the *binary cross-entropy* loss function and it is optimized using the *RMSProp* algorithm [124]. Hereafter, we refer to the proposed algorithm as LSTM-AD.

6.1.10 Performance Evaluation

The LSTM-AD algorithm is evaluated for different values of W and D. In particular, W represents the length of the observation window, which is equivalent to the number of lags of the stacked LSTM architecture. Instead, D indicates the number of parameters collected from the DCI messages that we need to process for the detection of the anomalies. We have tested all the possible combinations of the 5 parameters in the DCI messages described in Section 6.1.4. The objective of this study is to find the minimum value of D and W for which we obtain the highest accuracy. Finally, we also compare our solution with other state-of-the-art anomaly detection algorithms.

The performance tests have been carried out on cloud environment using Google Colaboratory, which provides free hardware acceleration with Tensor Processing Unit (TPU). The input dataset is split into training and validation sets with a ratio of 70% - 30% before the replication step. The anomaly detection algorithms have been implemented in Python: we use `keras` library and Tensorflow, as backend, to implement the LSTM-AD algorithm, while for the unsupervised algorithms we use the implementation from [125]. In the next section, we evaluate the results of the anomaly detection system, by defining, first, the evaluation metrics.

Performance Metrics

Defining proper metrics to evaluate the performance in an AD problem is fundamental: in most cases, in a multi-class classification problem, the *accuracy* (measured as the number of corrected classified samples over the total number of samples) is enough to explain the algorithm performance. However, when the classes are formed by an unbalanced number of samples, like in our case, the accuracy is not sufficient to evaluate the algorithm, since a blind classification of all the samples as normal behaviour can lead to a very high result. For these reasons we introduce additional metrics, namely *precision*, *recall* and *F-score*:

- **Precision** P: defined as the ratio between true positives T_p (the number of samples belonging to that anomaly class that are correctly classified) and the sum between true positives and false positives F_p, where F_p represents those normal samples that are incorrectly classified as anomalous,

$$P = \frac{T_p}{T_p + F_p} \tag{6.1}$$

- **Recall** R (also known as *sensitivity* or *hit-rate*): defined as the ratio between the true positives T_p and the sum between true positives and false negatives F_n, which are the anomalous samples incorrectly classified as normal, it gives the probability of detection of an anomalous behaviour,

$$R = \frac{T_p}{T_p + F_n} \tag{6.2}$$

- **F-Score** F is defined as the harmonic mean of precision P and recall R,

$$F = \left(\frac{\frac{1}{P} + \frac{1}{R}}{2}\right)^{-1} = 2\frac{RP}{R+P}. \tag{6.3}$$

LSTM-AD Algorithm Evaluation

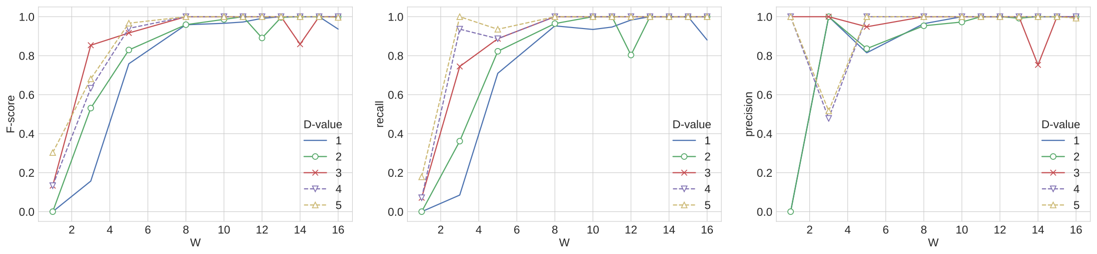
In Fig. 6.6, we use the Principal Component Analysis (PCA) to produce the 2D-scatter plots of the raw data (before it is input into LSTM-AD) and of the hidden features that are extracted by the last LSTM layer, before the FC softmax layer. We observe that a linear transformation like PCA is not able to separate the anomalies from the normal samples and justify the use of LSTM for our problem. In fact, the features extracted by the LSTM stacked architecture can definitely facilitate the estimation of a decision function to separate the two classes.

Fig. 6.7 gathers the performance results using the metrics that we previously defined for the anomaly class. The algorithm is evaluated for different values of W and D. First, we notice that the *precision* metric is not sufficient to evaluate the algorithm alone, since there are almost zero false-positive in the detection. Instead, from the *F-score* plot, we can observe that we obtain an F-score $F = 1$ when $W = 8$ and $D = 3$, meaning that we need to consider only the information about the number of C-RNTI and about the transport block size $[\text{RNTI}, \text{TBS}_{\text{down}}, \text{TBS}_{\text{up}}]$. Table 6.1 reports the results for the best input combination for D varying from 1 to 5 and $W = 8$. As shown in Fig. 6.7 and Table 6.1, increasing the dimensionality D it is not necessary, since the information given by the number the resource blocks allocation ($[\text{RB}_{\text{down}}, \text{RB}_{\text{up}}]$, $D = \{4, 5\}$) is implicitly included in the number of transport block size assigned.

6.1.11 Comparison with state-of-the-art AD

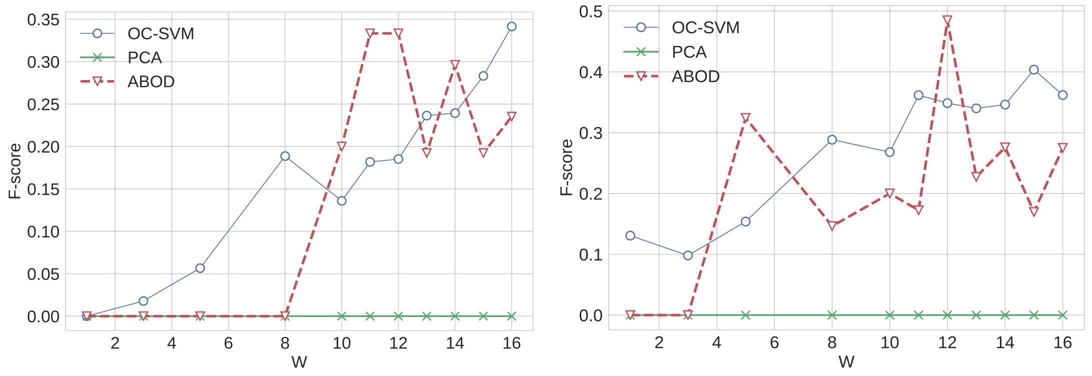
The anomaly detection with non-supervised algorithms is solely based on intrinsic properties of the data instances. The advantage is that they do not need the explicit labeling of the input data. Instead, their approach is to learn only the characteristics of the normal class and the classification of the anomalies is performed by comparing the new sample characteristics with the learnt characteristics.

We choose the following 3 methods as examples of non-supervised AD algorithms:



(a) **(b)** **(c)**

Figure 6.7: F-score, recall and precision of the LSTM-AD algorithm for different values of D and W. For each D, the choose the best input combination, as reported in Table 1.



(a) **(b)**

Figure 6.8: F-score obtained with OC-SVM, PCA and ABOD for different D and W values.

- One Class-SVM (OC-SVM) [126] is one of the most common one-class AD algorithms, and it is an extension of the Support Vector Machines to the AD problem;
- Angle-Based Outlier Detection (ABOD) [127] calculates the variance in the angles between the difference vectors of a point to the other points;
- Principal Component Analysis (PCA), with respect to the two former algorithms, is mostly used for feature selection and dimensionality reduction, but a variant of the PCA has been implemented and used in [128], for solving different outlier detection problems.

For these algorithms, we use the implementation presented in [125]: OC-SVM requires a parameter ν, defined as the upper bound on the fraction of outliers. This parameter regulates the tradeoff between maximizing the margin and the number of normal data points within the decision boundary: as done in [126], we choose a small value for ν ($\nu = 0.1$), since in our case the fraction of outliers is 8%.

In Fig. 6.8, we show the F-score of benchmark algorithms applied to our problem. As expected from Fig. 6.6a, the PCA cannot help distinguish the anomalies from the normal samples, achieving the poorest results in terms of F-score. On the other hand, OC-SVM and ABOD get positive F-score values with a maximum of 0.4 and 0.5 for $W = 15$ and $W = 12$, $D = 3$, respectively, which are much lower compared to the performances obtained with the LSTM-AD. This analysis proves that non-supervised algorithms are an alternative for the AD problem, in case you consider an unlabeled dataset, but they cannot reach the performance of the supervised approach, when a labeled dataset is available, as in the present work.

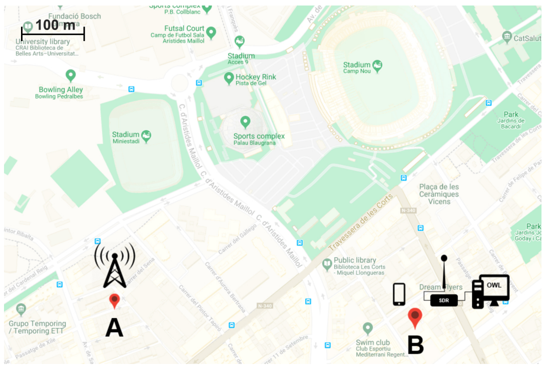
6.2 Detecting Anomalies through a Deep Semi-supervised Approach

6.2.1 Context and Motivations

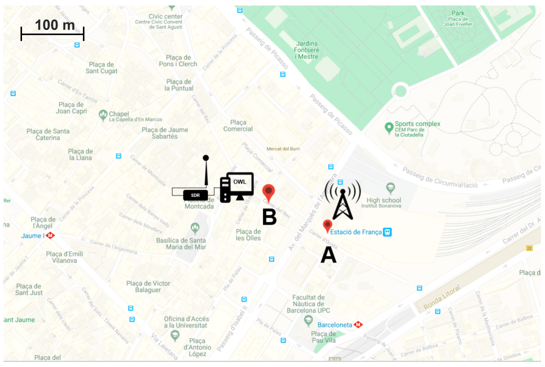
In this Section, we analyze data collected from an operative LTE network in Spain to perform mobile traffic anomaly detection. We focus on traffic that can present **contextual anomalies**, which are classified not only by their absolute values but also based on a specific temporal context. For example, a period of high traffic would be correctly classified as non-anomalous at regular peak times, but it would be an anomaly at low traffic hours (e.g. during the night).

After creating the dataset, we use a *semi-supervised* approach to train deep learning algorithms and detect the contextual traffic anomalies. With such methodology we overcome the so called *unbalanced class problem* [129], where one class is poorly represented with respect to the other. As a result of this, AD problem is not addressed as a supervised classification task, but rather, algorithms are semi-supervisedly taught to detect traffic anomalies learning only from non-anomalous samples.

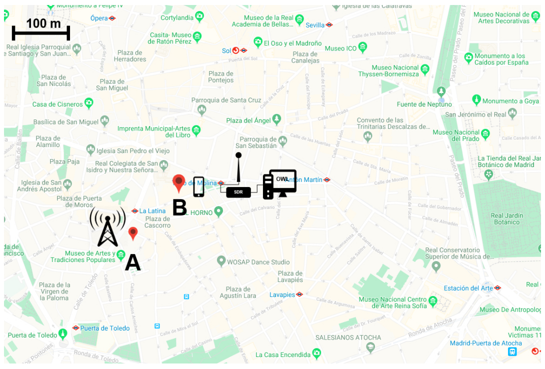
We address the problem of anomalies identification using RNN, which are one of the state-of-the-art deep learning techniques to deal with temporally correlated data. In particular, we use LSTM cells to build different deep learning architectures. For this, we design and evaluate two different approaches. In the first approach, we use LSTMs to implement an Autoencoder, which has the task of learning to reconstruct the normal input samples. In the second design, we exploit LSTM neural networks to predict the traffic at the next time-instant. The AD is then performed comparing the reconstruction and the prediction error against the groundtruth. A discussion on the performance of the algorithms and a comparison with shallow implementations is provided. The achieved results show the capabilities of the proposed deep learning framework to accurate detect the anomalies in the traffic data that are associated to different urban event typologies.



(a) Barcelona-Camp Nou



(b) Barcelona-Born



(c) Madrid-Rastro

Figure 6.9: Maps of Barcelona and Madrid where the measurement campaign took place for the creation of the dataset. In the maps, the eNodeB location is denoted by A, whereas the data collection system and the mobile terminal are marked as B.

6.2.2 Exploratory Data Analysis

The measurement campaign consists of about one month of data collection process for each eNodeB that takes place over operative mobile networks in Spain. The monitored eNodeBs are located in the two largest Spanish cities: Barcelona and Madrid. In this work, we include data corresponding to three eNodeBs located in different areas of these two cities. The maps in Fig. 6.9 show the exact locations of eNodeB where the sniffers have been placed for the duration of the measurements period.

The choice of the eNodeBs to be monitored is made based on the high variability of the mobile traffic in the selected areas. For each eNodeB, we identified a set of events occurred in the monitored period that gathers a variable number of people in the considered area and generate an abnormal mobile traffic demand. Information on the chosen events are publicly available [130, 131, 132]:

1. **Barcelona - Camp Nou**: the eNodeB is placed nearby the popular Camp Nou football stadium in Barcelona. The stadium is located in a urban residential area of Barcelona, which is characterized by a high population density. *Event:* The stadium is the largest in Europe and during football matches, it can host

up to 100K attendees;

2. **Barcelona - Born**: this area is a downtown district with a mixed residential, transport and leisure land use. The main activities include many restaurants and bars. *Event:* The measurement period includes Easter holidays and religious celebrations take place in the nearby area.

3. **Madrid - Rastro**: the eNodeB is located in centre of Madrid. The area is surrounded by the commercial activities that are either restaurants or small shops and the crowd in the surroundings is mainly pedestrian or slowly moving vehicles. *Event:* A periodic flea market known as "El Rastro" takes place in this area weekly, gathering a larger number of people.

For each area, we mark the respective associated event, and we exclude them from the set of data that we use to train the algorithms for the AD problem.

Dataset Structure

To study the total traffic exchanged between the eNodeB and all the connected users, we need to aggregate the eNodeB traffic from the PDCCH data. Let $\mathcal{T}$ be the total measurement period (see Table 6.2) for every second $t \in \mathcal{T}$ and define $\boldsymbol{x}(t)$ as the vector that contains the following information

1. RNTI: the total count of assigned C-RNTI;
2. RB_{down}: the total number of RBs allocated in the Downlink (DL) direction;
3. RB_{up}: the total number of RBs allocated in the Uplink (UL) direction;
4. MCS_{down}: the average MCS index assigned in the DL communication;
5. MCS_{up}: the average MCS index assigned in the UL communication;

We indicate with $D \in [1, 5]$ the number of features we consider in $\boldsymbol{x}(t)$. Therefore, the sequence $\boldsymbol{x}(t)$ is a multi-variate time-series, which includes the metrics that are extracted directly from the decoded DCI messages.

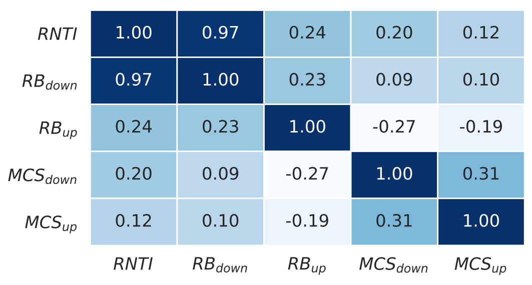
Dataset Visualization

Understanding the nature of the collected data and identifying significant patterns is fundamental in any data-driven approach. To this end, we perform an Exploratory Data Analysis (EDA) on the gathered mobile traffic data, to infer useful insights that can help to solve AD problem.

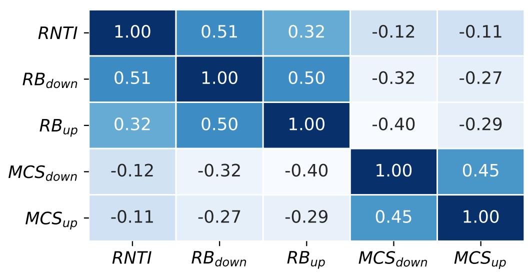
As first step, in Fig. 6.11, we show the average traffic profiles over 24 hours of the three considered eNodeBs in a typical day: as example, we plot the distribution of DL communication metrics (RNTI, RB_{down} and MCS_{down}). In Fig. 6.11a

Table 6.2: Collected PDCCH Dataset Statistics.

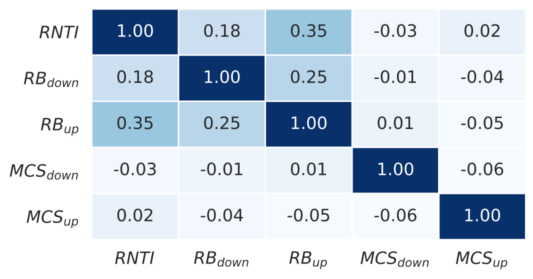
	Barcelona Camp Nou	**Barcelona Born**	**Madrid Rastro**
eNodeB bandwidth	20 Mhz	20 Mhz	10 Mhz
total PRBs	100	100	50
# of measurements	1002824	625278	1601292
measurement period $\mathcal{T}$	24 days	21 days	38 days
RNTI (avg±std)	4.70±3.27	79.2±62.9	4.8±3.1
RB_{down} (avg±std)	1.65±1.15	5.74±1.11	1.04±0.38
RB_{up} (avg±std)	0.47±1.8	0.70±2.2	0.86±.18
MCS_{down} (avg±std)	11.9±5.7	13.0±4.6	15.4±2.9
MCS_{up} (avg±std)	13.6±6.6	13.0±6.7	10.8±6.3



(a) Barcelona - Camp Nou



(b) Barcelona - Born

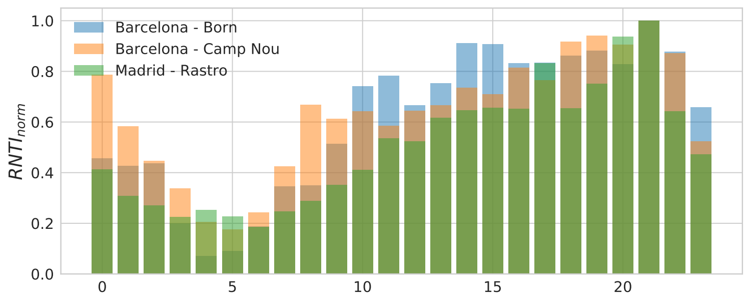


(c) Madrid - Rastro

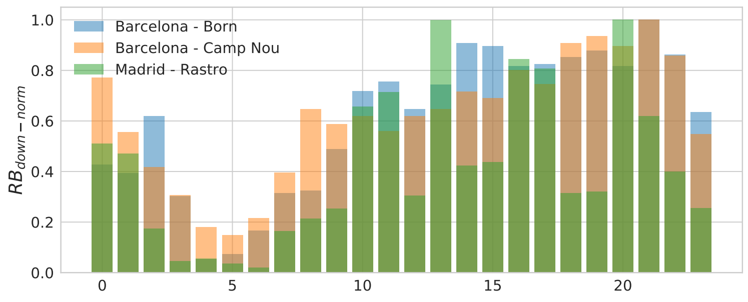
Figure 6.10: Pearson correlation matrix of the observed metrics.

and 6.11b, we can observe that the three traffic profiles present similar characteristics, but differ at some hour of the day. Night periods (12am - 7am) and day periods (8am - 11pm) are distinguishable and it is possible to observe that the traffic peak is reached around 8pm when typically residents are at the end of their working day. In Fig. 6.11a, RNTI values are maximum at 9pm and minimum at 5am. In Fig. 6.11b, for the Madrid eNodeB, we can observe that the RB utilization peaks are at 1pm and at 8pm, which corresponds to busy hour traffic, whereas it is minimum at 5am during non-busy hour period. In Fig. 6.11c, the average MCS index is around 15 for all the hours of the day (note that MCS can take values in [0,28]). These results show that the observed metrics are experiencing different behaviors including peaks and minimums at different times of the day.

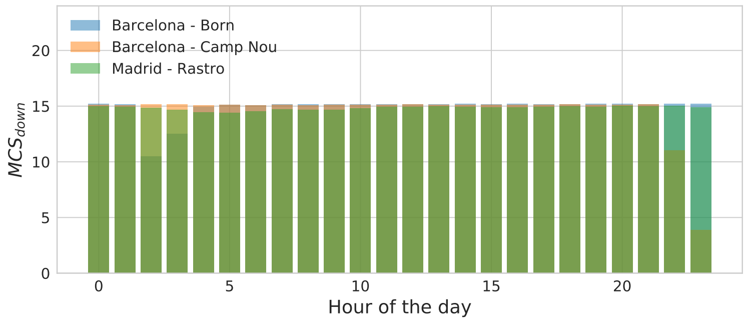
Fig. 6.10 shows the Pearson correlation between the variables of the collected datasets. We can observe that the correlations between the metrics of PDCCH data exhibit different values for each eNodeB. For example, in Barcelona-Camp Nou, RNTI and RB_{down} are highly correlated (with a value of 0.97), while in Barcelona - Born, RNTI and RB_{down} (RB_{up}) have values of 0.51 (0.32), respectively. This means that an increment in number of UEs in a given cell does not always correspond to an increase of the PRBs assignation. MCS correlation values are lower and, in general, the correlations between RNTI, MCS and RB are observed to be low, i.e. less than 0.5. The low values in the correlation matrix point out that there is not a straightforward dependence between the variables (for example between RB_{down} and RNTI). Therefore, we cannot further reduce the input dataset and also rather than focusing just on a single feature for AD, all the monitored metrics should be jointly utilized in mobile traffic anomalies identification problem.



(a) TBS Median



(b) RB Median



(c) MCS Median

Figure 6.11: Mean distributions for different KPI values over hours of day in downlink.

The heatmaps drawn in Fig. 6.12 indicate the median RNTI values per hour during one week at three different eNodeBs. For each eNodeB, the correspondent events are marked in red boxes:

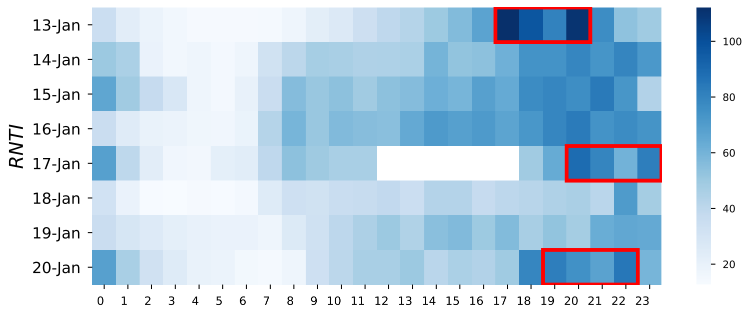
- In *Barcelona-Camp Nou*, the events correspond to football matches. The red box includes the start and the end of the match times with a offset of one hour. Three matches played at different hours are monitored (on 13th, 17th and 20th January, respectively). The most recognizable event is on January 13th, where a high value of RNTI was measured. However, detecting the other two matches is more difficult. For example, considering January's 17th no major changes occurred in the RNTI compared to the value measured on the previous days;
- In *Madrid-Rastro*, the registered event is the flea market that takes place weekly in the nearby area from 9am to 3pm. With respect to the Barcelona-Camp Nou dataset, the event is visually easier to be identified. However, this base station has only 10 Mhz bandwidth compared to the others (each with 20 Mhz bandwidth). Therefore, the absolute values of the difference of RNTI due to the event in this area is minor with respect to other two considered events. As example, we also plot the heatmaps of RB_{down} in Fig. 6.13b for comparison purposes. Due to low correlation between the metrics and possibility to increase the detection rates for the considered AD problem, we have considered multiple metrics in the subsequent analysis;
- In *Barcelona-Born*, the measurements period took place during Easter week. The notable events during this period include the religious celebrations (on 29th, 30th March, and 1st April at 12pm). In particular, on April 1st (Easter day), two celebrations occurred: one around 10pm the night before (Easter Eve) and the other one at 12pm.

6.2.3 Semi-Supervised Framework

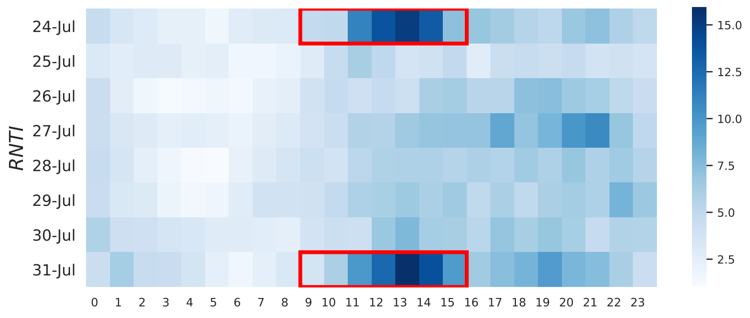
The general framework for solving the AD problem is depicted in Fig.6.14, which is named as LSTM-AD. This framework comprises a limited number of pre-processing steps on the collected raw data to minimize the detection time of potential anomalies. To accomplish this, the framework first takes the data collected from LTE PDCCH as input. Later, it essentially performs three parts: *A) Data Preprocessing*, *B) AD Algorithm Learning* and *C) AD Decision Function*. The implementation details of each parts are in turn discussed in the next sections.

Data Preprocessing

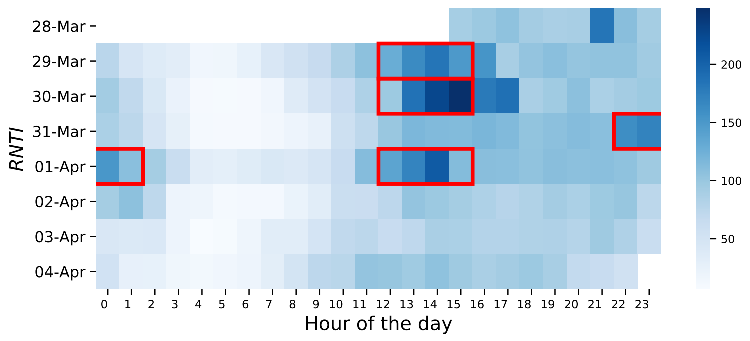
Before being input to the AD algorithms, the mobile traffic dataset is preprocessed. Given the multi-variate mobile traffic time-series $\boldsymbol{x}(t)$, we perform the following two preprocessing steps:



(a) Barcelona - Camp Nou.



(b) Madrid - Rastro.

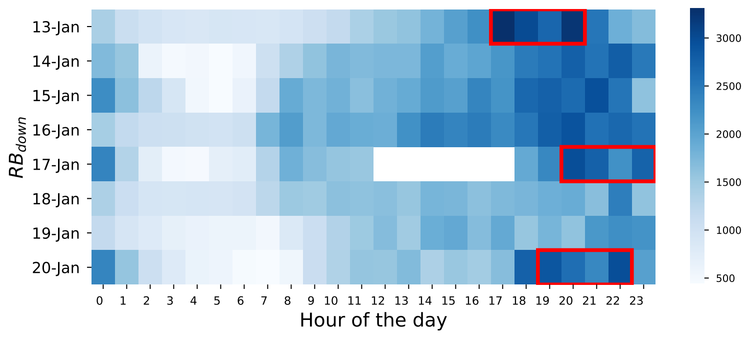


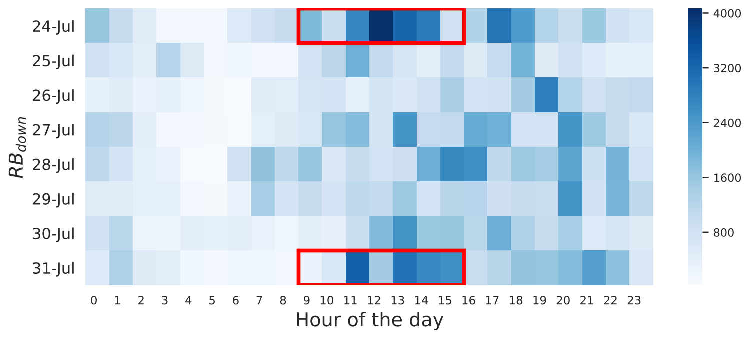
(c) Barcelona - Born.

Figure 6.12: Heatmaps of one week of RNTI values collected from three different eNodeBs where chosen events are marked in red boxes.

- *Data sampling and normalization*: The sequence $x(t)$ is resampled using a value t_s and standardized by removing the mean and scaling to unit variance. This operation is performed to filter and normalize the original sequence to reduce the variance of the input dataset (in our experimental results, the plots are obtained with $t_s = 30$). Hereafter, to simplify the notation usage, we also use $x(t)$ to indicate the resampled sequence.

- *Data windowing*: The sequence $x(t)$ is split and grouped using a fixed-length window W. The window is moved each time by one-step. The value of W defines how many time-lags are processed by the AD-algorithm that classifies if the input is an anomaly or not.


(a) Barcelona - Camp Nou.


(b) Madrid - Rastro.

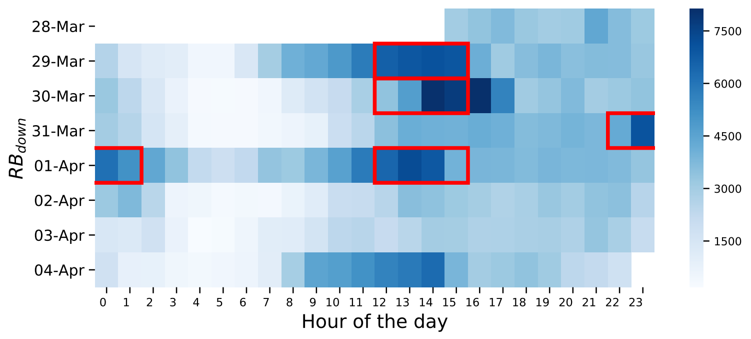

(c) Barcelona - Born.

Figure 6.13: Heatmaps of one week of RB_{down} values collected from the eNodeBs where chosen events are marked in red boxes.

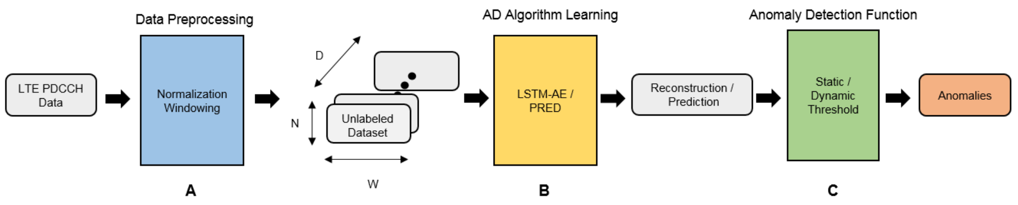


Figure 6.14: LSTM-AD Framework: A general framework to solve AD problem.

The multi-variate sequence can be expressed as $[x(1), x(2), \ldots, x(T)]$, where T the cardinality of $\mathcal{T}$, i.e. $T = |\mathcal{T}|$. After the split, we have $N = T - W + 1$

sequences $\mathbf{x}(n), n \in [1, N]$

$$\mathbf{x}(1) = [\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(W)]$$

$$\mathbf{x}(2) = [\boldsymbol{x}(2), \ldots, \boldsymbol{x}(W+1)]$$

$$\mathbf{x}(N) = [\boldsymbol{x}(N), \ldots, \boldsymbol{x}(T)]$$

A sequence $\mathbf{x}(n)$ has length W and each of its element is D-dimensional, with $D \in [1, 5]$, the number of considered metrics as described in Section 6.2.2. Hereafter, we refer to the sequence $\mathbf{x}$ as *samples*. Then, we can define $\mathbf{X}$ the three-dimensional matrix which contains N sequences of $\mathbf{x}$. The matrix $\mathbf{X}$ has dimension $N \times W \times D$ and serve as input *tensor* to the AD algorithm.

AD Algorithm Learning

We design two neural network architectures, namely Autoencoder and Predictor, to automatically extract the relevant features from the LTE PDCCH dataset and we train them to reconstruct or predict the normal traffic instances. The AD is achieved by studying the reconstruction or prediction error, which is supposed to be higher for anomalous traffic instances.

Specifically in our implementation, we use *stacked-LSTM* neural networks, which are the state-of-the-art deep learning structures to deal with sequential data. LSTMs are capable of learning long-term dependencies from the input time series, while solving the vanishing-gradient problem that affects standard RNNs [133]. This capability is due to the structure of the basic LSTM cells (or units) that includes gates to regulate the learning process.

LSTM Autoencoder the general approach using an LSTM-Autoencoder (LSTM-AE) is depicted in Fig. 6.15. Generally, autoencoders are used in representation learning to learn unsupervisedly a representation of the input in a feature space. In our case, we implement a *sequence-to-sequence* autoencoder [134], since our dataset consists of time-series sequences. The objective is to reconstruct the traffic samples using an *encoded representation* of the input sequence.

An autoencoder consists of an encoder and a decoder. Let $\mathcal{X} = R^D$ be the input space and $\mathcal{F}$ be the feature space. An *encoder* is a function $\phi : \mathcal{X} \to \mathcal{F}$ that has the task of learning the prominent characteristics and creating an encoded version of the sample in the feature space $\mathcal{F}$. The *decoder* instead is a function $\psi : \mathcal{F} \to \mathcal{X}$ that aims to reconstruct the input using the internal representation.

For the implementation of both the encoder and the decoder, we use LSTM cells, which do not consider independent inputs with respect to plain neurons (or perceptrons) and are capable of extracting the temporal dependencies from one instance

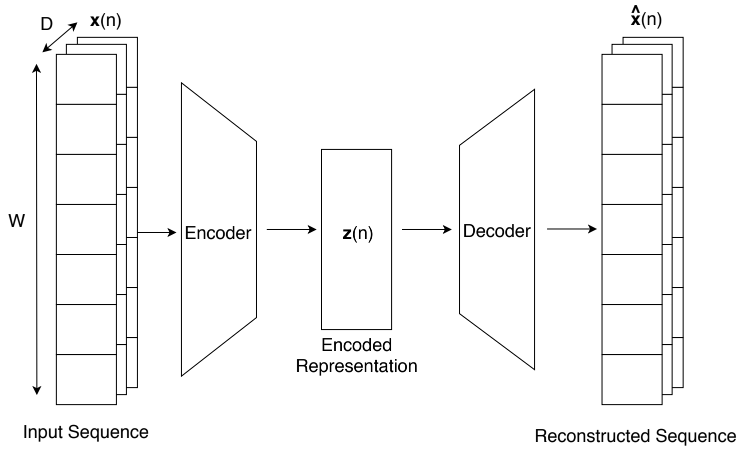


Figure 6.15: LSTM Autoencoder for Anomaly Detection.

and another. To address the AD problem, the idea is to train the autoencoder in a *semi-supervised* way using only traffic samples without potential anomalies. Formally, given a sample sequence $\mathbf{x}(n)$, the autoencoder is a function $\Phi_{AE} : \prec \circ \leftarrow$ that outputs $\hat{\mathbf{x}}(n)$

$$\Phi_{AE}(\mathbf{x}(n)) = \hat{\mathbf{x}}(n) \tag{6.4}$$

With sufficient training samples, the architecture is taught to learn to reconstruct the normal samples with a *low reconstruction error* compared to anomalous events.

LSTM Prediction we can use the LSTM architecture to make traffic prediction as well (LSTM-PRED, see Fig. 6.16). Therefore, instead of trying to reconstruct the input samples, the objective now becomes to predict the traffic in the next time-instants. By definition, anomalies are unlikely predictable, therefore, the idea is that the algorithm is taught to predict only the traffic in regular conditions, where the prediction error is expected to be low.

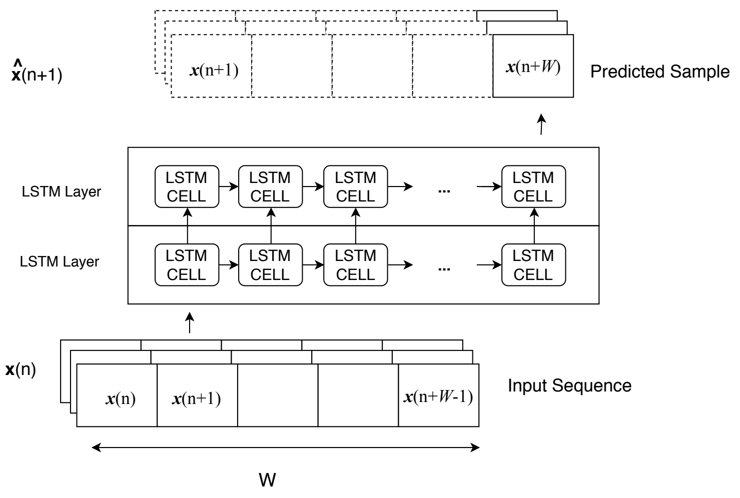


Figure 6.16: LSTM Architectures used for Anomaly Detection.

As input, the algorithm (LSTM-PRED) receives a traffic sequence of length W at time n, $\mathbf{x}(n) = [x(n), x(n+1), .., x(n+W-1)]$, and tries to predict the traffic sample at time $n+W$, $\hat{x}(n+W)$

$$\Phi_{PRED}(\mathbf{x}(n)) = \hat{x}(n+W) \tag{6.5}$$

Similar to LSTM-AE, we train the LSTM-PRED using only regular traffic samples. Our implementation consists of a stacked architecture that includes multiple LSTM layers. The number of concatenated cells in the first layer indicates the number of observations of the data, which in our case corresponds to the window length W.

AD Decision Function

Given the output from the AD algorithm learning block, the objective of this part is to evaluate the reconstructed (or predicted) traffic and decide if it has to be classified as anomalous or not. First, we calculate the Mean Square Error (MSE) between the reconstructed (or predicted) sequences and their true values averaged over the D metrics:

$$\mathcal{L}_{AE}(n) = \frac{1}{W}\frac{1}{D}\sum_{W}\sum_{D}|\Phi_{AE}(\mathbf{x}(n)) - \mathbf{x}(n)|^2 \tag{6.6}$$

$$\mathcal{L}_{PRED}(n+W) = \frac{1}{D}\sum_{D}|\Phi_{PRED}(\mathbf{x}(n)) - x(n+W)|^2 \tag{6.7}$$

The classification of anomalies is performed as *inference* on the input traffic sequence. Both algorithms are trained only on normal traffic samples, hence the errors produced for anomalous sequences are expected to be higher, making the algorithm being able to classify them as anomalies.

Instead of setting a static error threshold, we can use the first and the second order statistics to implement the decision function that classifies an error as anomaly, and we can calculate a *dynamic* error threshold that take account of the different traffic behaviours during the different hours of the day.

We accomplish this using *moving average with discrete linear convolution* method. Similar to [135], the moving average $\mu(n)$ is calculated as a linear convolution between the error $\mathcal{L}_{AE/PRED}(n) = err(n)$ and a low-pass filter $\mathcal{K}(n)$ with length W. Then, we calculate σ as the standard deviation of the residual. If the absolute difference between the error and the moving average is greater than σ, the correspondent traffic instance is marked as anomaly (see pseudo-code in Algorithm 2).

Algorithm 2 AD Decision Function

```
1: procedure ANOMALY DETECTION
   ▷ calculate moving average using kernel $\mathcal{K}$
2:     $\mathcal{K}$: low-pass filter
3:     $\mathcal{A}$: set of anomaly samples
4:     $\mu(n) \leftarrow err(n) * \mathcal{K}(n)$
5:     $residual \leftarrow err(n) - \mu(n)$
6:     $\sigma \leftarrow$ standard deviation $(residual)$
   ▷ mark the anomalies
7:     if $|err(n) - \mu(n)| > \sigma$ then
8:         mark sample at time $n$ as anomaly
9:         add $n$ to set of anomaly samples $\mathcal{A}$
10:    return $\mathcal{A}$
```

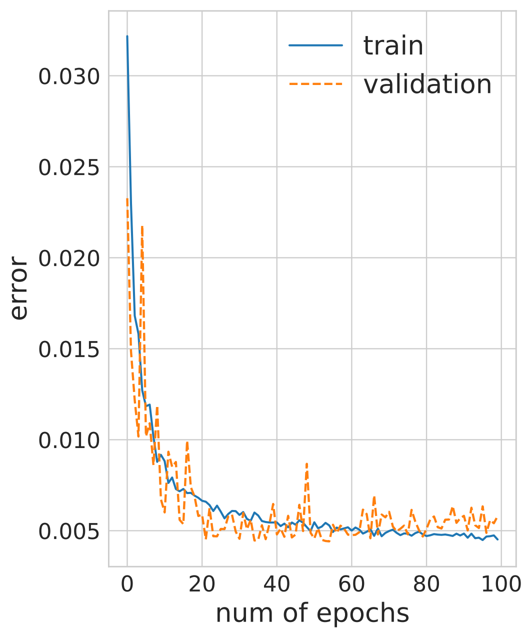
6.2.4 Performance Evaluation

The two proposed algorithms are evaluated in terms of *precision*, *recall*, *F-score* and *accuracy* metrics. Note that accuracy alone is not sufficient to demonstrate the capabilities of algorithms due to the class unbalance problem, in which even a blind classification of all the samples as normals can lead to a very high accuracy results.

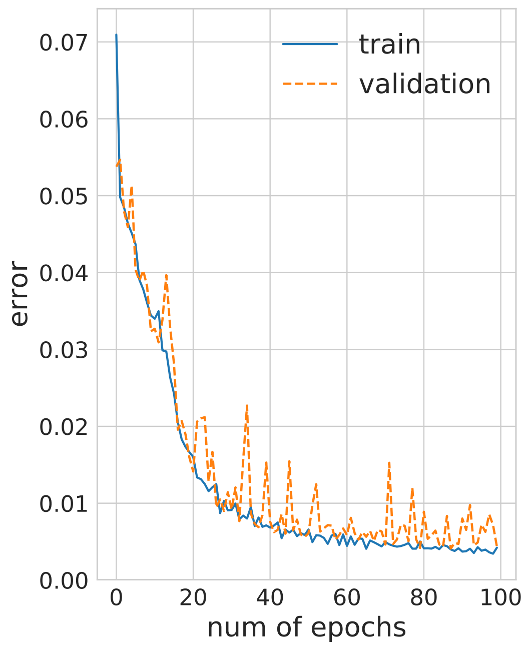
Training phase

Before training the algorithms, first we divide the dataset that is composed of only normal samples (we exclude the measurements that take place during notable events) into training, validation and test sets, using a split ratio of $\sim 0.50, 0.25, 0.25$. We train and validate the algorithms on the training (or validation) sets to minimize the reconstruction error (in the case of LSTM-AE) or prediction error (in the case of LSTM-PRED). The test set is used at inference time for anomaly prediction and includes the weeks when the events occured: for Barcelona-Camp Nou dataset, the test set is the week between 13th and 20th January, for Barcelona-Born, is the week between 28th March and 4th April and for Madrid-Rastro, it includes two weeks between 15th July and 2nd August.

In Fig. 6.17, we plot the train and validation errors versus the number of epochs during the training phase for the LSTM-PRED (left) and LSTM-AE (right). The parameters used for the training are chosen by grid-search validation in order to obtain an MSE ≤ 0.01 (reported in Table 6.3). For LSTM-AE, we use $N_{L_{AE}} = 2$ LSTM layers and $H_{AE} = 100$ hidden neurons for implementing both encoder and decoder. In LSTM-PRED, we consider a stacked architecture combining $N_{L_{PRED}} = 2$ LSTM hidden layers with $H_{PRED} = 100$ hidden units each. The error is calculated as MSE and the *RMSProp* algorithm is used to optimize the learning process. As can be observed from Fig. 6.17, the error values start to saturate after ~ 50 epochs.



(a) LSTM-PRED.



(b) LSTM-AE.

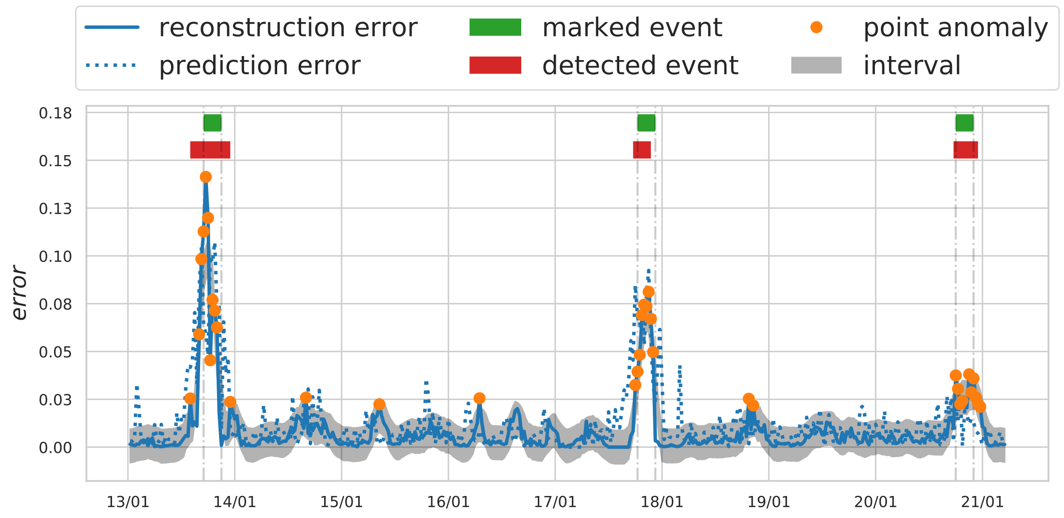
Figure 6.17: Training and Validation error vs. number of epochs.

Table 6.3: Configuration of learning parameters.

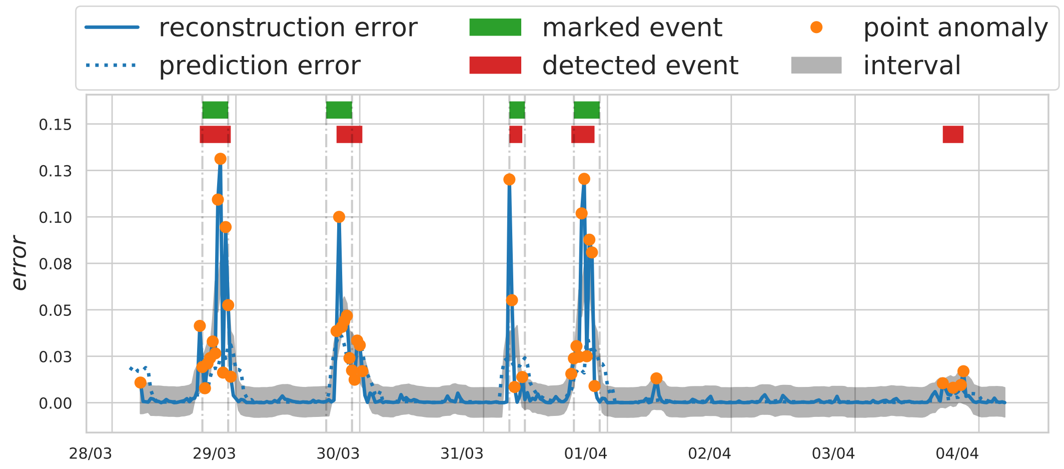
	Parameters	**Value**
$N_{L_{PRED}}$	number of layers in LSTM-PRED	2
$N_{L_{AE}}$	number of layers in LSTM-AE	2
H_{PRED}	number of hidden LSTM cells in each layer in LSTM-PRED	100
H_{AE}	number of hidden LSTM cells in each layer in LSTM-AE	100
W	moving-window samples	6
D	number of features	5
Opt	optimization algorithm	$RMSprop$

6.2.5 Anomaly Detection

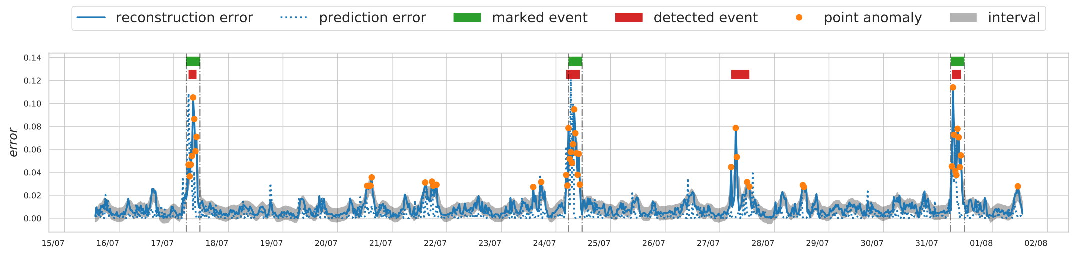
The anomaly identification is achieved at inference time using the test set. We evaluate the algorithm performance by measuring the reconstruction error in the case of LSTM-Autoencoder (LSTM-AE) and the prediction error in the case of LSTM-PRED as given in equations (6.6) and (6.7) respectively. We also use the decision function defined in Sec. (6.2.3) to classify if the considered traffic instance is an anomaly or not.



(a) LSTM-AE used on Barcelona-Camp Nou dataset.



(b) LSTM-PRED used on Barcelona-Born dataset.



(c) LSTM-AE used on Madrid-Rastro dataset.

Figure 6.18: AD results on the test datasets of the three different datasets collected from three eNodeBs.

In Fig. 6.18, we present three plots (one for each eNodeB) that show the proposed algorithm's error performance and the correspondent detected anomalies over the test dataset. We also mark the periods when the chosen events took place. We can distinguish two types of traffic anomalies: the contextual anomalies (which are associated to the identified events) and point anomalies (which we do not associate to any specific urban event).

Event Identification

Based on the decision function, in Fig. 6.18 we label the anomalies so that they can be compared with the marked events (in green mark) that serve as ground-truth. In this case, since we are interested in identifying contextual anomalies, we can select the periods that present subsequent anomalies (red mark). This allows us to identify specific events and gives a characterization of such events including the duration, the starting time and the ending time.

Some of the observations are as follows: first, in Fig. 6.18 the error is observed to be higher in correspondence to the marked events (e.g., when the football games are played) and *all* the marked events are successfully detected by the proposed algorithms. Second, Fig. 6.19 shows the obtained results with the two semi-supervised algorithms using accuracy, precision, recall and F-score as metrics. Considering that the training is done using only the normal samples, for both algorithms the *precision* is very high (> 0.9 on average). However, the *recall* is lower (0.6 on average), since the detected events are slightly shifted with the respect to the marked periods as observed in Fig. 6.18. To understand these differences, it is appropriate to refer to every single case:

- In Barcelona-Camp Nou, due to the high capacity of the stadium (about 100K attendees), people start gathering in the area nearby the stadium few hours before the match kick-off. Consequently, the mobile traffic exchanged at the correspondent eNodeB increases before the game starts;
- In the Madrid-Rastro dataset, the period between the opening and closing hours of the flea market are marked where this event takes place weekly from 9am to 3pm. However, even if the market starts at 9am, the majority of people concentrates around 12pm, making the detected events slightly to be shifted to the right. In the performance evaluation, this has the effect of lowering the recall values, since few false-negatives are produced;
- In Barcelona-Born, we observe multiple events taking place on consecutive days. This is because of Holy Week during Easter holiday. The events occur at different times, but they are linked to religious celebrations organized in the nearby area. In particular, we can observe that all the events take place at day time, except the one that is shown on March 31st, which corresponds to Easter eve. It traditionally occurs at night between the Holy Saturday and the Easter Day.

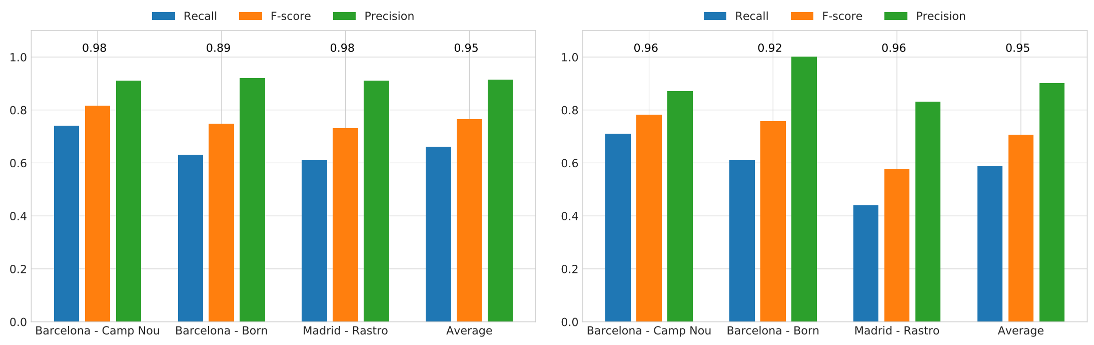
Other Type of Anomalies

The proposed methodology is also general enough to detect potential traffic anomalies that are not preliminarily marked and that are not associable to the monitored events. In fact, we can observe from Fig. 6.18 that few *point anomalies* are not comprised between the marked events. Point anomalies are generally associated to an *isolated* occurrence, for example, due to an instant increase of the traffic demands by the users, and are not related to any urban event.

In our case, point anomalies are also observed in all the three monitored eNodeBs. We can observe both isolated phenomena and subsequent anomalies that are marked as detected events. These affect the precision of the proposed algorithms, lowering the overall F-score. In particular, we observe that

- in the Madrid-Rastro dataset, we detect traffic anomalies on July 27th, which is a week day that cannot be associated to the flea market event. No similar events have been reported in this commercial area, however, this day corresponds to the longest total lunar eclipse of the 21st century [132], which we believe had gathered a crowd of interested people to observe this uncommon phenomena;
- in Barcelona-Born, we identify anomalies on April 4th: observing the heatmaps of Fig. 6.12 and 6.13, we can see that these anomalies are not recognizable from the RNTI heatmap figure, but an increase of RBs in the DL direction (RB_{down}) can be observed. However, the detected traffic anomaly may be associated to some liturgics's celebration in the week after Easter (known as Octave of Easter).

Based on these results, we assert that the proposed algorithm allows for a general identification of crowd-gathering anomalies, not restricted to any particular type of event. This ability is due to the semi-supervised learning methods, which permits to the algorithm to separate the regular traffic instances from the ones related to unusual events.



(a) Performance measures of LSTM-AE.

(b) Performance measures of LSTM-PRED.

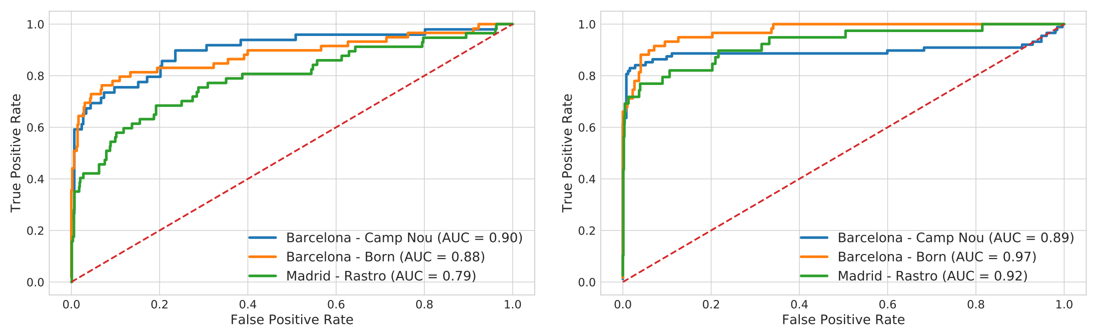
Figure 6.19: Performance measures of the AD algorithms where a static AD function is applied. F-score is reported on top of the bars.

Considerations

To compare the classification performance of the proposed algorithms, we also plot in Fig. 6.20 the *Receiver Operating Characteristic* (ROC) using the prediction and reconstruction errors from LSTM-PRED and from LSTM-AE. For each algorithms, we calculate the *Area Under Curve* (AUC), which represent a degree of separability between the two classes. For LSTM-AE, the results are similar for all the datasets with an average AUC of 0.93. For LSTM-PRED the performance are inferior (average AUC = 0.85), in particular for Madrid-Rastro, where the True Positive Rate (TPR) (or recall) value is observed to be lower than other eNodeBs as a consequence of detecting some anomalies that are not correlated with the marked events.

Overall, the performance of LSTM-AE is observed to be higher compared to LSTM-PRED. This is due to the fact that reconstructing an entire anomaly sequence with LSTM-AE results more complex than predicting a singular traffic sample. In fact, the autoencoder fails to reconstruct a traffic sequence that differs too much from the normal sequences that is learnt from during the training phase. Therefore the error becomes larger and easier to be identified in the test dataset. Moreover, the error increases when a sequence contains more than one point anomaly. On the other hand, the high ability of LSTM cells to predict even irregular patterns makes the prediction error limited also for traffic with anomalies.

The tradeoff between having a precise traffic prediction and an accurate anomaly identification should be evaluated in a realistic operative scenario. In fact, LSTM-PRED can be used *online* to perform multiple task: traffic prediction and anomaly detection. If the next instant traffic load is very large, there is a probability that it may be associated to an anomaly. This makes the algorithm being able to detect potential anomalies almost in *real-time*.



(a) LSTM-PRED. **(b)** LSTM-AE.

Figure 6.20: Receiving Operation Characteristic of the AD algorithms (a) LSTM-PRED (b) LSTM-AE.

Table 6.4: Results and references of anomaly detection on different datasets using state-of-the-art algorithms.

Dataset	Algorithm	Ref	Prec	Rec	F-Score	Acc
Camp Nou	K-Nearest Neigh	[136]	0.29	0.78	0.42	0.90
	One Class-SVM	[126]	0.25	0.77	0.36	0.89
	Isolation Forest	[137]	0.27	0.74	0.39	0.89
Born	K-Nearest Neigh	[136]	0.44	0.78	0.56	0.87
	One Class-SVM	[126]	0.45	0.76	0.57	0.88
	Isolation Forest	[137]	0.44	0.76	0.56	0.87
Rastro	K-Nearest Neigh	[136]	0.31	0.90	0.46	0.91
	One Class-SVM	[126]	0.30	0.92	0.46	0.91
	Isolation Forest	[137]	0.29	0.91	0.44	0.90

6.2.6 State-of-the-art Comparison

For the sake of completeness, in this section, we present the comparison between the proposed deep learning algorithms and three standard semi-supervised AD benchmarks: k-Nearest Neighbours [136], One-Class SVM [126] and Isolation Forest [137].

- *k-Nearest Neighbours* (k-NN): is a classic non-parametric distance-based method that are used for both classification and regression purposes. In classic k-NN, the learning is mostly supervised. However in its enhanced version for AD problems, k-NN is trained in a semi-supervised manner, with only one class of samples [136]. The detection of anomalies occurs at inference when a new instance is projected far from the normal cluster in the feature space;

- *One Class-Support Vector Machine* (OC-SVM): is one of the most common single-class AD algorithms, and is an extension of the SVMs to the AD problem [126]; OC-SVM requires a parameter ν, defined as the upper bound of the fraction of outliers. This parameter regulates the tradeoff between maximizing the margin and the number of normal data points within the decision boundary;

- *Isolation Forest*: is an ensemble tree-based method, that depends on the spatial proximity of the observed samples. The algorithm isolates the observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The recursive partitioning can be represented by a tree structure and the number of splits required to isolate a sample is equivalent to the path length from the root node to the terminating node. If a sample requires multiple splits to be isolated, it is likely that it will be classified as anomalous.

For all these above benchmarks, we used PyOD, which is a scalable implementation for detecting outlying objects in multivariate data presented in [125]. For k-NN, we use $k = 5$. For OC-SVM, we choose a small value for ν ($\nu = 0.1$) (which represents the upper bound of the fraction of outliers) since in our case the fraction of outliers is 8%. In the Isolation Forest, we use the default number of estimators ($n_{tree} = 100$).

The results are reported in Table 6.4 and a comparison with the F-score of the proposed deep learning algorithms is given in the barplot of Fig. 6.21. We can observe that the recall value is very high (in particular, in the Madrid-Rastro dataset), with an average of 0.81 versus 0.61 of the proposed algorithm. However, the precision (0.34 vs. 0.89), and consequently the F-score (0.47 vs. 0.72), are much lower.

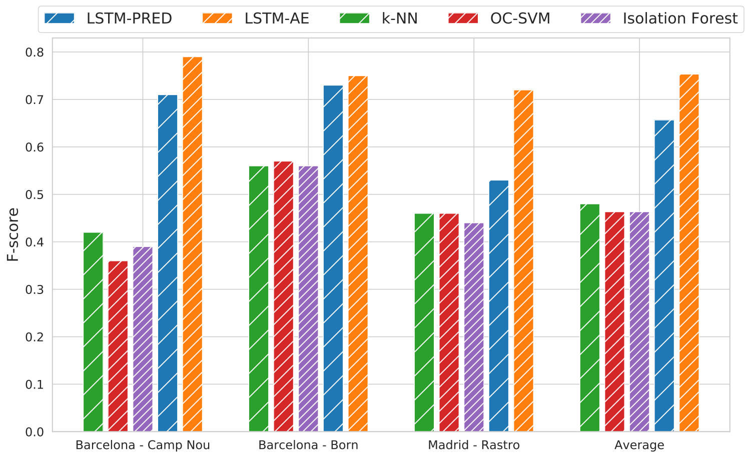


Figure 6.21: Barplots of the algorithms F-score on the different dataset.

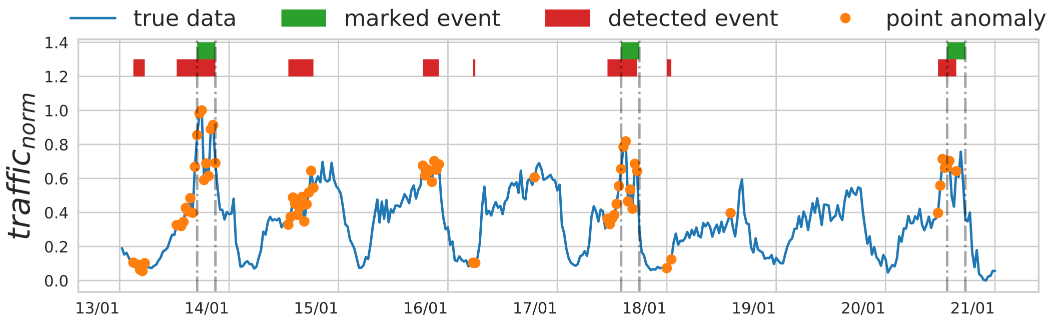


Figure 6.22: Anomaly detection with Isolation Forest applied to the Barcelona-Camp Nou dataset: numerous false-positives are observed.

To understand this outcome, we plot the normalized traffic in Fig. 6.22: we can see that most of the peak traffic hours are marked as anomalies, thus increasing the number of false positives. This means that even the normal traffic instances are classified as anomalies. The main conclusions are that these traditional algorithms, differently from our approach, detects all the traffic peaks, including the expected daily maximums, but are not able to differentiate them from the traffic generated by crowd-gathering events. This explains that the F-score is lowered by 34% on average with respect to the proposed algorithms.

Chapter 7

LTE Channel Fingerprinting for Traffic Classification and Prediction

This Chapter provides a methodology for the classification and the prediction of mobile traffic through the LTE Physical Channel fingerprinting: in Section 7.1, we present a viable methodology to identify the demanded services and apps only by monitoring the unencrypted control information channel and we are able to classify the users with very high accuracy. The results of this section are presented in the following paper:

- Trinh, H. D., Gambin, A. F., Giupponi, L., & Dini, P. (2019). **Mobile Traffic Classification through Physical Channel Fingerprinting: a Deep Learning Approach**. *arXiv preprint arXiv:1910.11617.* [24]

Next, in Section 7.2, we present an application of the predictive model introduced in Section 5.3 and of the classification algorithm proposed in Section 7.1 in the context of energy saving devices. Together with authors of [25], we propose a wake-up scheme to enhance the energy-efficiency of 5G mobile devices is proposed in order to prolong the battery lifetime, while reducing the buffering delay. The results reported in this Section are presented in the following papers:

- Rostami S., Trinh, H. D., Lagen S., Costa M.,Valkama M. & Dini, P. (2019). **Proactive Wake-up Scheduler based on Recurrent Neural Networks**. *arXiv preprint arXiv: 1910.11617.* [25]

7.1 Classification of Mobile Services and Apps using LTE Physical Channel Information

With the advent of powerful chipsets, high-definition bezel-less screens, and upgraded memory storages, mobile devices have become information and communication hubs and are getting more powerful by the day. Starting with 4G, mobile applications (apps) are as well getting resource hungry, requiring large bandwidth and good amount of computation at the devices. The demand for streaming application is increasing at a constant pace, for example, more than half of YouTube views come from mobile terminals [138, 2], and future networks are called to be highly responsive in any context, e.g., high-speed mobility, indoor, crowded events, etc. [139].

A key feature towards supporting such technological evolution, and the corresponding upsurge in multimedia traffic, is represented by the ability of networks to carry out automatic traffic analysis, through *online classification tools*. The categorization of network traffic into appropriate classes has many relevant uses spanning from Quality of service (QoS)/Quality of Experience (QoE) control and management, to pricing, network resource management, malware detection, and intrusion detection, to name a few. The key challenge of such classification algorithms consists in the identification, and in the subsequent computation, of a number of *representative features*. These features are then used to train algorithms that classify the data flows at runtime. Most of the surveyed approaches leverage some *domain knowledge*, which is utilized to *manually* obtain the feature set, i.e., using prior information by a human expert. However, the use of deep learning techniques has recently paved the way to automatic feature discovery and extraction, often leading to superior performance. For example, in [83] encrypted traffic is categorized through deep learning architectures, proving their better performance with respect to shallow neural network classifiers. The authors of [84] present a mobile traffic super-resolution technique to infer narrowly localized traffic consumption from coarse measurements. In detail, a deep-learning architecture combining Zipper Network (ZipNet) and Generative Adversarial neural Network (GAN) models is put forward to accurately reconstruct spatio-temporal traffic dynamics from measurements taken at low resolution. Another example is found in [85], where identification of mobile apps is performed by automatically extracting features from labeled packets through CNNs, which are trained using raw HTTP requests, achieving a high classification accuracy. The work in [83, 84, 85], as the majority of the other techniques, use statistical features obtained from application or IP level information for both service and app identification, along with UDP/TCP port numbers.

7.1.1 Objectives and Methodology

The solution presented in this Chapter sharply departs from prior approaches as it performs highly accurate traffic classification directly from *radio-link* level data, without requiring any prior knowledge and without having to decode and/or decrypt the transmitted data flows. The proposed classifiers enable fully automated, over the air, and detailed traffic profiling in mobile cellular systems, making it possible to characterize the radio resource usage of typical service classes. To this end, we leverage OWL [16], a tool that allows decoding the LTE PDCCH, where control information is exchanged between the LTE eNodeB and the connected UEs. Specifically, we take advantage of the DCI messages carried in the PDCCH, which contain radio-link level settings for the user communication (e.g., modulation and coding scheme, transport block size, allocated resource blocks). From DCI data, we create two datasets:

1) a *labeled dataset*, used to *train* service and app classification algorithms, where labeling is made possible by injecting an easily identifiable watermark into the application flows generated by a terminal under our control;

2) an *unlabeled dataset*, used for *traffic profiling* purposes, which is populated by monitoring, for a full month, mobile traffic from four operative radio cell sites with different demographic characteristics within the metropolitan area of Barcelona, in Spain.

For the traffic analysis, we focus on those services and applications that dominate the radio resource usage, but the approach can readily be extended to further services and applications. We directly use raw DCI data as input into deep learning classifiers (automatic feature extraction), achieving accuracies as high as 99% for both mobile service and app identification tasks. Moreover, taking advantage of the high accuracy of our CNN classifier, we propose a technique to successfully use it in unsupervised settings, to profile the mobile traffic from operative radio cell sites at runtime. Our tool allows for a *fine grained* and *automated* analysis of user traffic from real deployments, using radio link messages transmitted over the PDCCH. The developed classification algorithms, as well as our experimental results are highly novel within the traffic monitoring literature, which only provides hourly and *aggregated* measures for typical days [140][141], and where traffic profiling is performed from UDP/TCP, IP or above IP flows, e.g., [83, 84, 85].

In summary, the original contributions of this work are:

- *Mobile Data Labeling:* we present an original and effective approach to automatically label LTE PDCCH DCI data traces. This approach is utilized for six mobile apps, to create a unique correspondence between the software programs (the apps) and the session identifiers that were assigned to them by the eNodeB. The result is a *labeled dataset* of real DCI data from selected applications, i.e., YouTube, Vimeo, Spotify, Google Music, Skype and WhatsApp video calls.
- *Classification and Benchmarks:* we tailor deep artificial Neural Networks (NNs), namely Multi-Layer Perceptron (MLP), RNNs and CNNs, to perform classification tasks for both mobile services and app identification over the labeled dataset. Moreover, we compare their performance against a number of benchmark classifiers, based on state-of-the-art supervised learning algorithms. CNN architectures achieve the highest classification accuracy.
- *Mobile Service Profiling from Unlabeled Data:* the CNN classifier, which is found

to be the best among all NN schemes, is augmented with the capability of rejecting out of distribution sessions, i.e., sessions whose statistical behavior departs from those learned during the training phase. This makes it possible to use the CNN classifier with unlabeled traffic, in an online fashion. In fact, the augmented CNN classifier rejects those sessions for which it is uncertain, providing a robust classification outcome. Through its use, daily traffic profiles for the four monitored eNodeBs are obtained, getting a fine grained decomposition of the services requested by the active users. The proposed augmentation strategy exploits theoretically sound and lightweight techniques, and constitutes an innovative contribution of this thesis.

7.1.2 Data Collection System

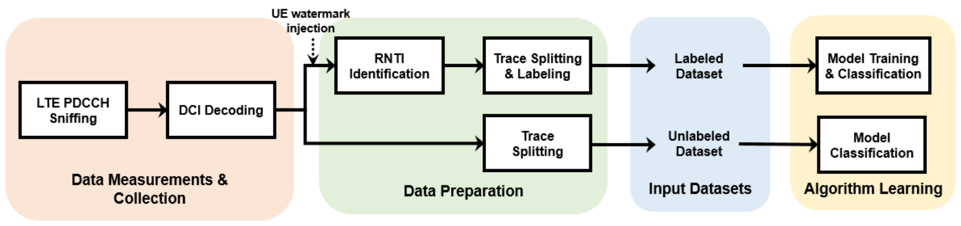


Figure 7.1: Experimental framework adopted for the creation of the unlabeled and labeled datasets.

Dataset Creation

Fig. 7.1 shows the different building blocks of the experimental framework that has been developed to populate the unlabeled and labeled datasets. Briefly, the *data measurement and collection* block acquires data from the LTE PDCCH channel to extract the relevant DCI information. *Data preparation*, instead, processes the gathered DCI data so that it can be used for training and classification purposes, according to the type of dataset.

In LTE, the eNodeB communicates scheduling information to the connected UEs through the DCI messages that are carried within the PDCCH with a time granularity of 1 ms. While the actual user content is sent over *encrypted dedicated channels*, i.e., the Physical Uplink/Downlink Shared Channel (PUSCH/PDSCH respectively), the **PDCCH is transmitted in clear text** and can be decoded. To process DCI data, we have adapted the OWL monitoring tool [16]. A SDR has been programmed, acquiring the PDCCH via an open-source software sitting on top of the srs-LTE library [110], which makes it possible to synchronize and monitor the channel over a specified LTE bandwidth. The SDR is connected to a PC that performs the actual decoding of DCI data: in our experimental settings, we used a low cost Nuand BladeRF x40 SDR and an Intel mini-NUC, equipped with an i5 2.7 Ghz multi-core processor, 256 GB Solid State Storage (SSD) storage and 18 GB of RAM.

Unlabeled Dataset

Thanks to the tool described in the previous section, four cell sites of a Spanish mobile network operator in the metropolitan area of Barcelona have been monitored for a full month. The selected eNodeBs are located in areas having different demographic characteristics and land uses, so as to diversify the captured traffic in terms of service and app behavior. We have named the datasets according to the corresponding neighborhood: *PobleSec* (mainly residential area), *Born* (mixed residential, transport and leisure area), *Castelldefels* (mixed suburban and campus area), *Camp Nou* (mixed residential and stadium area). In total, we have collected more than 68 GB of DCI data from the LTE PDCCH. Fig. 6.9 shows the locations of the four monitored sites, along with that of the data collection system. After the data collection, the signaling associated with each found C-RNTI is extracted from the PDCCH DCI data stream, and is prepared for the classifier. During this procedure, we discard short-length traces, which are mainly due to signaling, paging and background traffic. Such data accounts for less than 3% of the total traffic in the monitored radio cells.

Labeled Dataset

A *labeled* dataset is obtained by running specific services and apps at a mobile terminal under our control, detecting the identifier of that terminal within the PDCCH channel and finally associating the corresponding DCI trace with a *label*, which links it to the service/app that is executed at the UE. Generating data sessions is easy, as it is sufficient to run a specific app from a device that we control, and that is connected to the monitored eNodeB. The difficult part is to identify the generated data flow among those carried by the PDCCH channel, which contains DCI information for *all* the connected UEs within the radio cell. We made this labeling possible by injecting a *watermark* into the traffic that we generate by the controlled UE, so that the latter can be easily identified among all other users, as we now explain.

C-RNTI Identification

The data preparation procedure is divided into two phases: **1)** the identification of the C-RNTI corresponding to the controlled UE, **2)** the extraction and labeling of the corresponding DCI trace. In the LTE PDCCH channel, each UE is identified by the C-RNTI, which uniquely identifies the mobile terminal within the radio cell. The identifier though is *temporary*, i.e., it changes after short inactivity periods. This is due to prevent the plain tracking of mobile users, since the PDCCH is sent unencrypted. To overcome this, we introduced a *watermark* into the traffic generated through our mobile terminal. This watermark amounts to producing, for each application, a regular pattern: any given application (e.g., YouTube) is run for a pre-defined amount of time (60 s is used in our measurements), then, a pause interval of fixed duration is inserted before running the app for further 60 seconds.

We loop this over time, obtaining a duty cycled activity pattern that is easily distinguishable from all the other activity traces within the radio cell. Fig. 7.2 shows an example of this watermark injection procedure: the traffic generated by our UE is represented through a blue solid line, for which the duty cycled activity is evident and easily identifiable from that of other users in the cell (dashed orange line).

Through this watermarking approach, the DCI data associated with our UE can be easily obtained from the PDCCH channel, allowing us to track the corresponding C-RNTI as a function of time. The label, corresponding to the application that is being executed at the mobile terminal, is finally associated with the extracted DCI data. In our experiments, DCI traces from the same app are split into labeled sessions of 60 s each. This method guarantees a high UE identification accuracy and can be automated, allowing the collection of thousands of labeled DCI traces.

In our measurement campaign, we have recorded and labeled about $M = 10,000$ mobile sessions, gathering the scheduling information contained in the DCI messages for several apps. We considered three data-intensive services: *video streaming*, *audio streaming* and *real-time video calling*, which represent classes producing a considerable amount of traffic and taking most of the network resources [138]. For each service type, we chose two popular applications: *Spotify* and *Google Music* for audio, *YouTube* and *Vimeo* for video streaming, while for the video calling we picked two instant-messaging applications, namely, *Skype* and *WhatsApp Messenger*.

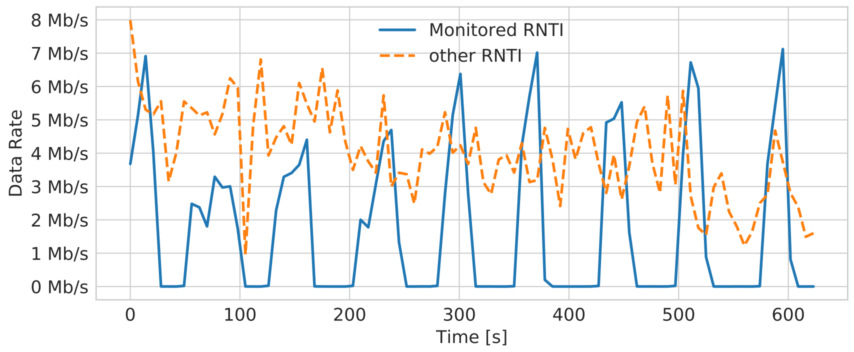


Figure 7.2: Example audio traffic *vs.* unknown session traces with (solid line) and without (dashed line) watermark: the regular pattern allows the identification of the user, i.e., of the assigned C-RNTI.

A large measurement campaign has been conducted to expose the mobile terminal running the selected apps to different radio link conditions and obtain a comprehensive dataset. In particular, the UE has been placed into two different locations (termed B_1 and B_2 in Fig. 4.1a) within the Castelldefels radio cell to experience different received signal qualities (-84 dBm and -94 dBm for B_1 and B_2, respectively), and in the Camp Nou eNodeB during football matches, to capture data in high cell load conditions.

Fig. 7.3 shows a few radio resource usage patterns collected for the selected apps. Some similarities can be recognized within the same service class. For example, audio and video streaming present similar behaviors. Also, significant differences can be observed between the radio resource usage of real time video calls (*Skype* and *WhatApp Video*) and the other apps. Video and audio streaming applications use up a high amount of radio resources at the beginning of the sessions, buffering most of the content into the terminal memory. Real time video calling, instead, entails a continuous transmission and a more constant usage of radio resources throughout the sessions.

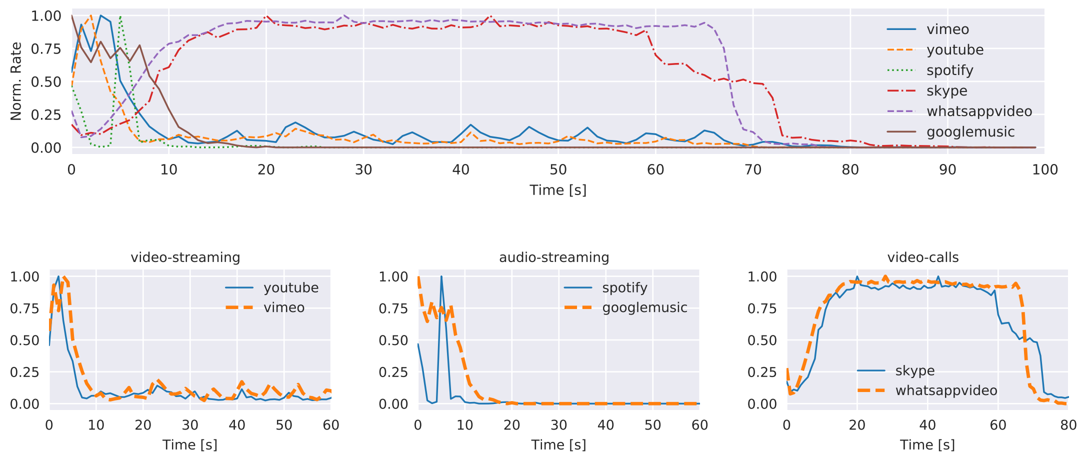


Figure 7.3: Traffic pattern snapshots showing the normalized data rate for different applications as a function of time.

7.1.3 Synchronous and Asynchronous Sessions

Through the watermarking approach and the splitting procedure, we obtained a labeled dataset, where each session, depending on the service, presents patterns similar to those shown in Fig. 7.3. Assuming that the beginning and the end of each session are known is rather optimistic, as in a real measurement setup we have no means to accurately track these instants. Put it another way, it is unlikely that the LTE PDCCH measurements and the application run on the UE will be temporally *synchronized*. Synchronizing the measurement with the beginning of each session would facilitate the classification task, since most of the generated traffic is buffered on the terminal at the beginning, see Fig. 7.3, and this behavior is a distinctive feature that is easy to discriminate.

To ensure the applicability of our classifiers to real world (asynchronous) cases, we account for *asynchronous* sessions, entailing that the classification algorithm has no knowledge about the instants where the sessions begin and end. Specifically, each session is split using a sliding window of length W seconds, moved rightwards from the beginning of the session with a stride of S seconds, see Fig. 7.4. The split sessions (asynchronous sessions), of W seconds each, represent the input data to our

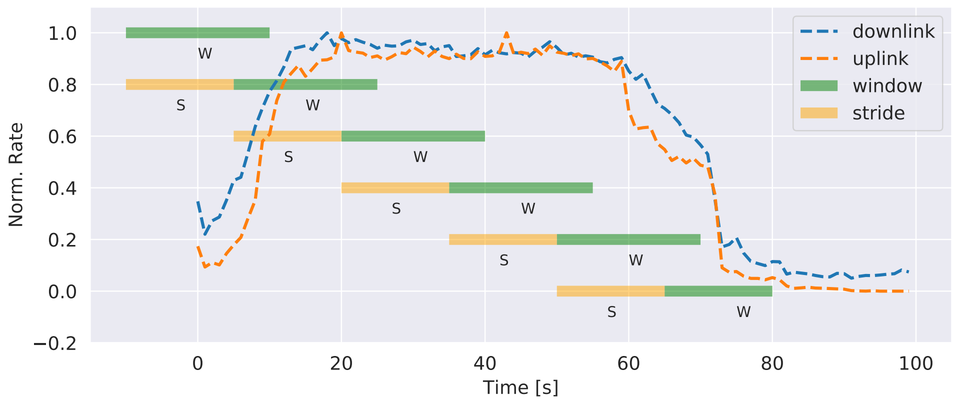


Figure 7.4: Sliding window of 20s length and 15s stride, applied to a sample video-streaming session.

classification algorithms. Note that W and S are hyper-parameters of the proposed classification frameworks.

7.1.4 Sessions Correlation over Time

As a sanity check, we verify the soundness of the watermarking strategy: our aim is to understand whether the transmission of user data in the form of duty-cycled patterns may affect the way in which the eNodeB handles the communication from our terminal, e.g., through some advanced channel reservation mechanism. In that case, in fact, our watermarking strategy would be of little use, as it would introduce scheduling artifacts that do not occur in real life conditions. To verify this, we evaluated the Pearson correlation between the initial session (i.e., when the UE connects to the LTE PDCCH for the first time and it is assigned a new C-RNTI) and the following ones. Fig. 7.5 shows that, for each of the three services, the correlation is high only when we compare the first session with itself ($n = 0$). Instead, low values are observed between the first session and the following ones ($n > 1$), indicating that the behavior of the eNodeB scheduler is not affected by the repetitive actions (i.e., the duty-cycled activity) performed at the UE side.

7.1.5 Problem Statement

Let M be the number of windowed-sessions obtained through the data preparation procedure of Section 7.1.2, W is the window size, and $D = 2$ is the number of communication directions (downlink and uplink). We define $\boldsymbol{X}$ the input dataset tensor with size $M \times W \times D$, where the m-th row vector $\boldsymbol{x}_m$ contains the trace associated with W TBS samples per session for both downlink and uplink directions ($D = 2$).

A classifier estimates a function $c : \boldsymbol{X} \to \boldsymbol{Y}$, where matrix $\boldsymbol{Y}$ has size $M \times K$ and K represents the number of classes. The row vector $\boldsymbol{y_m} = c(\boldsymbol{x_m}) = [y_{m1}, \ldots, y_{mK}]$ contains the probabilities that session m belongs to each of the K classes, with $\sum_k y_{mk} =$

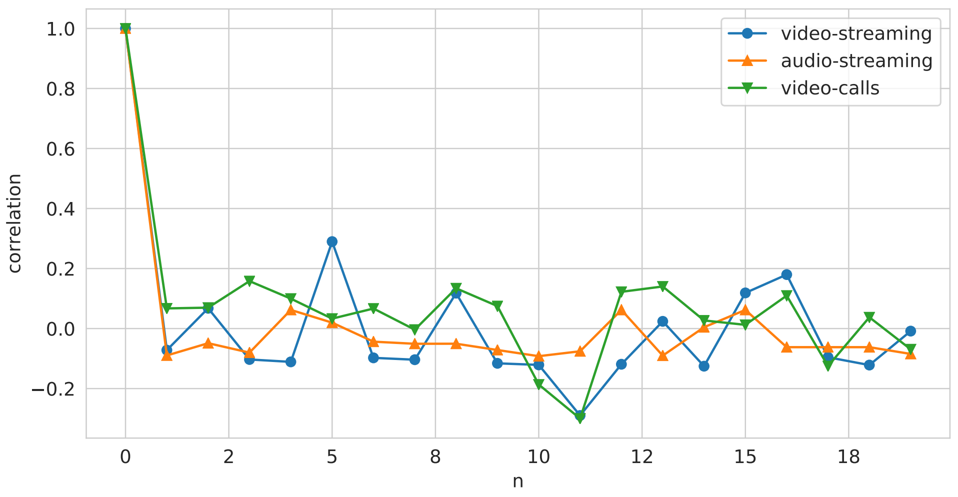


Figure 7.5: Pearson correlation between the initial and the following sessions running in the controlled UE.

1. The final output of the classifier is class $k^\star$, where $k^\star = \text{argmax}_k(y_{mk})$. The following classification objectives are addressed:

O1) **Service identification:** to classify the collected sessions into $K = 3$ classes, namely, *audio streaming*, *video streaming* and *video calls*;

O2) **App identification:** to identify which app is run at the UE. In this case, the number of output classes is $K = 6$, namely, *Spotify*, *Google Music*, *YouTube*, *Vimeo*, *Skype* and *WhatsApp Messenger*.

Next, we present the considered classification algorithms, grouping them into two categories: those based on artificial neural networks and the others on 'standard' machine learning techniques (referred to here as benchmark classifiers).

7.1.6 Neural Networks for Classification

In this section, we describe how we tailored three neural network architectures to solve the above traffic classification problem. The considered architectures are Multilayer Perceptron (MLP), RNNs and CNNs.

Multilayer Perceptron

A multilayer perceptron is a feedforward and fully-connected neural network architecture. The term "feed-forward" refers to the fact that the information flows in one direction, from the input to the output layer. An MLP is composed of, at least, three layers of nodes: an input, a hidden and an output layer. A directed graph connects the input with the output layer and each neuron in the graph uses a non-linear activation function to produce its output. Links are weighted and the backpropagation algorithm is utilized to train the network in a supervised fashion, i.e., to find the set of network weights that minimize a certain error function, given an input set of examples and the corresponding labels. For further details, see [142].

The MLP that we use for mobile traffic classification has *three* fully connected layers. The input layer MLP_1 contains $N_{MLP_1} = 128$ neurons, the second layer MLP_2 has $N_{MLP_2} = 64$ neurons and the third layer MLP_3 is fully connected, with K neurons and a softmax activation function to produce the final output. The output of MLP_3 is the class probability vector $\boldsymbol{y_m}$.

All neurons in layers MLP_1 and MLP_2 use a leaky version of the Rectified Linear Unit (ReLU) (*leaky ReLU*) activation function. Leaky ReLUs help to solve the vanishing gradient problem, i.e., the fact that the error gradients that are backpropagated during the training of the network weights may become very small (zero in the worst case), preventing the correct (gradient based) adaptation of the weights. To prevent this from happening, leaky ReLUs have a small negative slope for negative values of their argument [143]. To train the presented MLP architecture, we use the *RMSprop* gradient descent algorithm [124], by minimizing the *categorical cross-entropy loss function* $L(\boldsymbol{w})$, defined as [142]

$$L(\boldsymbol{w}) = -\sum_{\boldsymbol{x}_m \in \boldsymbol{B}} \sum_{k=1}^{K} t_k(\boldsymbol{x}_m) \log(y_{mk}(\boldsymbol{w}, \boldsymbol{x}_m)). \tag{7.1}$$

where $\boldsymbol{t(x)_m} = [t_1(\boldsymbol{x}_m), \dots, t_k(\boldsymbol{x}_m)]$ contains the class labels associated with the input trace $\boldsymbol{x}_m$, i.e., $t_k = 1$ if $\boldsymbol{x}_m$ belongs to class k and $t_k = 0$ otherwise (1-of-K coding scheme). Vector $\boldsymbol{w}$ contains the MLP weights and $y_{mk}(\boldsymbol{w}, \boldsymbol{x}_m)$ is the MLP output obtained for input $\boldsymbol{x}_m$. Eq. (7.1) is iteratively minimized using the training examples in the batch set $\boldsymbol{B} \subset \boldsymbol{X}$, where $\boldsymbol{B}$ is changed at every iteration so as to span the entire input set $\boldsymbol{X}$.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have been conceived to extract features from temporal (and correlated) data sequences. LSTM networks are a particular type of RNN, introduced in [103]. They are capable of tracking long-term dependencies into the input time series, while solving the vanishing-gradient problem that affects standard RNNs [102].

The capability of learning long-term dependencies is due to the structure of the LSTM cells, which incorporates gates that regulate the learning process. The neurons in the hidden layers of an LSTM are MCs. A MC has the ability to retain or forget information about past input values (whose effect is stored into the cell states) by using structures called *gates*, which consist of a cascade of a neuron with sigmoidal activation function and a pointwise multiplication block. Thanks to this architecture, the output of each memory cell possibly depends on the entire sequence of past states, making LSTMs suitable for processing time series with long time dependencies [103].

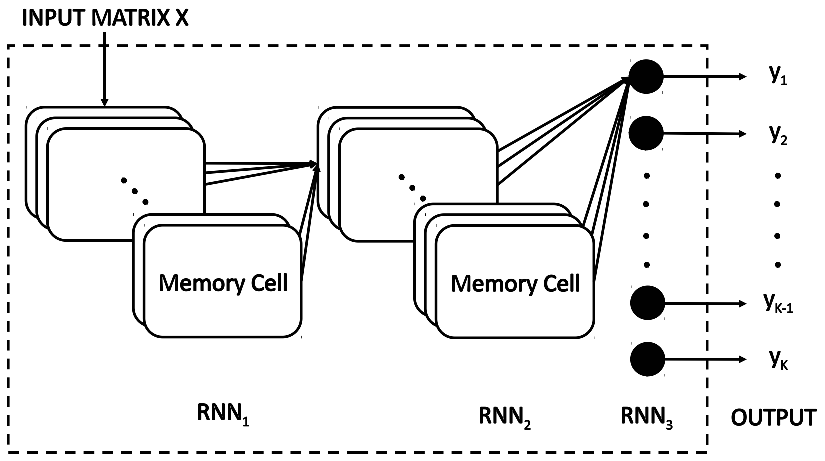


Figure 7.6: RNN architecture.

The proposed RNN based traffic classification architecture is shown in Fig. 7.6. In our design, we consider three stacked layers combining two LSTM layers and a final fully connected output layer. The first and the second layer (respectively RNN_1 and RNN_2) have $N_{RNN_1} = N_{RNN_2} = 180$ memory cells. The fully connected layer RNN_3 uses the softmax activation function and its output consists of the class probability estimates, as described in Section 7.1.6.

Convolutional Neural Networks

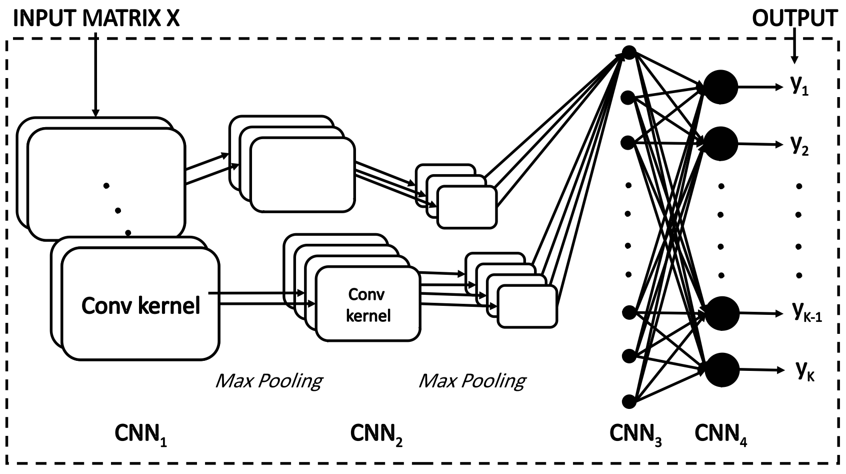


Figure 7.7: CNN architecture.

Convolutional Neural Networks (CNNs) are feed-forward deep neural networks differing from fully connected MLP for the presence of one or more convolutional layers. At each convolutional layer, a number of kernels is used. Each kernel is composed of a number of weights and is convolved across the entire input signal. Note that the kernel acts as a filter whose weights are re-used (*shared weights*) across the entire input and this makes the network connectivity structure sparse, i.e., a small set of parameters (the kernel weights) suffices to map the input into the output. This leads to advantages such as a considerably reduced computational complexity with respect to fully connected feed forward neural networks, and to a smaller memory footprint. For more details the reader is referred to [105].

CNNs have been proven to be excellent feature extractors for images and inertial signals [144] and here we prove their effectiveness for the classification of mobile traffic data. The CNN architecture that we designed to this purpose is shown in Fig. 7.7. It has two main parts: the first four layers perform convolutions and max pooling in cascade, the last two are fully connected. The first convolutional layer CNN_1 uses one dimensional kernels (1×5 samples) performing a first filtering of the input and processing each input vector (rows of $\boldsymbol{X}$) separately. The activation functions are linear and the number of convolutional kernels is $N_{CNN_1} = 32$. The second convolutional layer, CNN_2, uses one dimensional kernels (1×5 samples) with non-linear hyperbolic tangents as activation functions, and the number of convolutional kernels is $N_{CNN_2} = 64$. Max pooling is separately applied to the outputs of CNN_1 and CNN_2 to reduce their dimensionality and increase the spatial invariance of features [144]. In both cases, a one-dimensional pooling with a kernel of size 1×3 is performed. A third fully connected layer, CNN_3, performs dimensionality reduction and has $N_{CNN_3} = 32$ neurons with Leaky ReLU activation functions. This layer is was used in place of a further convolutional layer to reduce the computation time, with a negligible loss in accuracy. The last (output) layer CNN_4 is fully connected with softmax activation functions, and returns the class probability estimates, see Sections 7.1.6.

7.1.7 Benchmark classifiers

Other standard classification schemes have been tailored to the considered tasks O1 and O2. The selected algorithms are: *Linear Logistic Regression*, *K-Nearest Neighbours* and *Linear SVM*, as examples of linear classifiers; *Random Forest*, as an ensemble learning method, and *Gaussian Processes* as an instance of Bayesian approaches. The implementations of Linear Logistic Regression, K-Nearest Neighbours and Linear SVM are based on [145], [146] and [147], respectively. The Random Forest implementation is based on [148], whereas for the classifier based on Gaussian Processes we refer to [149]. Configuration parameters and implementation details for the benchmark classifiers are provided in Table 7.1.

7.1.8 Supervised Training and Comparison of Traffic Classifiers

Performance Metrics

The performance tests have been carried out using an Intel core i7 machine, with 32 GB of RAM and an NVIDIA GTX 980 GPU card. We divided the dataset, featuring

Table 7.1: Configuration parameters for the benchmark classifiers.

Algorithm	Parameters	Note - Reference
Linear SVM	• *penalty* = L2 • *loss* = Hinge Loss • $c = 0.025$	• c: penalty parameter for the error term • extended to multi-class with one-vs-rest [145]
Logistic Regressor	• *penalty* = L2 • $c = 1$	• c: inverse of the regularization strength • extended to multi-class with one-vs-rest [145]
Nearest Neighbours	• $K = 3$ • $p = 2$ • *metric* = Minkowski	• K: number of neighbors for queries • p: distance metric parameter • $p = 2$ amounts to using the Euclidean distance [146]
Random Forest	• *n. estimators* = 10 • *max depth* = 5 • *criterion* = entropy	• *n. estimators*: number of trees in the forest • *max depth*: maximum depth of a tree • *criterion*: function to measure the quality of a split of subsets [148]
Gaussian Processes	• *kernel* = RBF • σ = Logistic func. • *approx.* = Laplacian	• Radial Basis Function (RBF) used as kernel • σ is the sigmoid function used to "squash" the nuisance function • Laplacian method used to approximate the non Gaussian Posterior [149]

10,000 labeled DCI sessions, into training and validation sets with a split ratio of 70% - 30%. These sets are balanced, as they contain the same percentage of traces for all classes. The classification algorithms have been implemented in Python. We have used `keras` library on top of Tensorflow backend for the implementation of deep NNs. For the benchmark classifiers, we used the popular `sklearn` library.

The classification performance is assessed through the following metrics:

1. **Accuracy**: defined as the ratio between the number of correctly classified sessions to the total number of sessions.

2. **Precision** P: defined, for each class, as the ratio between the number of samples belonging to that class that are correctly classified (true positives T_p) and the sum between true positives and false positives F_p, where F_p represents those samples that are incorrectly classified as belonging to that class,

$$P = \frac{T_p}{T_p + F_p}, \tag{7.2}$$

3. **Recall** R: defined, for each class, as the ratio between the true positives T_p and the sum between true positives and false negatives F_n, which are the samples from that class that are incorrectly classified as belonging to another class,

$$R = \frac{T_p}{T_p + F_n}. \tag{7.3}$$

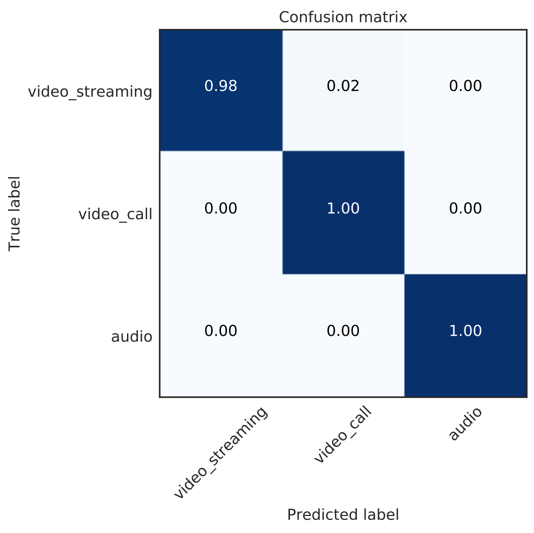
Algorithm	**Accuracy** %	**Precision**	**Recall**	**F-Score**	**# Parameters**	**Acc Async** %	**Diff** %
Linear SVM	81.23	0.811	0.812	0.805	36726	68.41	-12.8
Logistic Regressor	81.61	0.806	0.816	0.809	486	65.72	-15.9
Nearest Neighbours	84.51	0.843	0.845	0.841	36720	79.65	-4.9
Random Forest	83.52	0.821	0.835	0.827	41310	70.21	-13.3
Gaussian Processes	87.43	0.874	0.871	0.871	146720	81.21	-6.2
Neural Networks							
MLP	90.04	0.900	0.900	0.900	19014	84.61	-5.4
RNN	96.57	0.967	0.968	0.968	392046	92.93	-3.6
CNN	97.77	0.978	0.976	0.977	25062	93.20	-4.5

Table 7.2: Classifiers comparison for the app identification task.

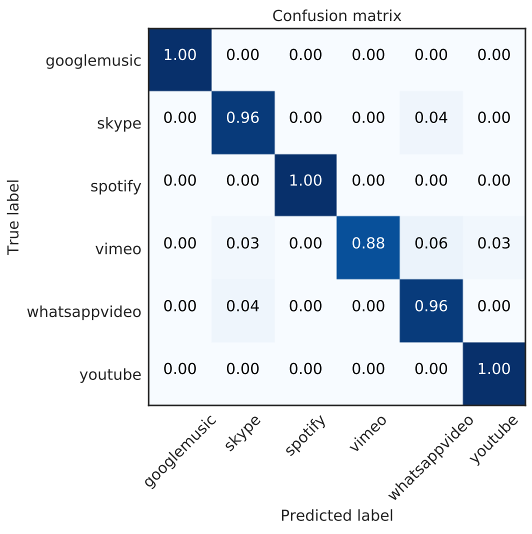
4. *F-Score* is defined as the harmonic mean of precision P and recall R,

$$F = \left(\frac{\frac{1}{P}+\frac{1}{R}}{2}\right)^{-1} = 2\frac{RP}{R+P}. \tag{7.4}$$

Note that the definition of precision and recall only applies to classification tasks with one class. However, tasks O1 and O2 both have a number of classes $K > 2$, namely, $K = \{3, 6\}$ for app and service identification, respectively. Thus, precision and recall are separately calculated for all the K classes, and their average is shown in the following numerical results.



(a) Confusion matrix for service identification.



(b) Confusion matrix for app identification.

Figure 7.8: Confusion matrices for the CNN algorithm.

7.1.9 Comparison of Classification Algorithms

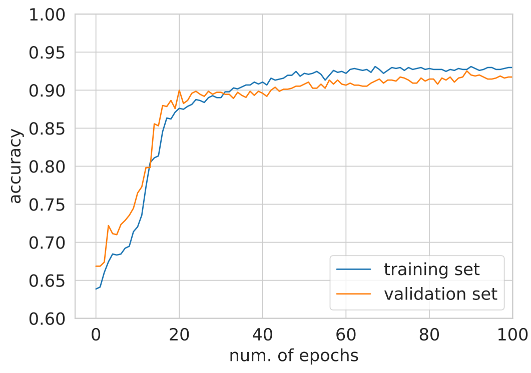
Accuracy and Algorithm Training

Tables 7.2 and 7.3 summarize the obtained performance metrics for the deep NNs and the benchmark classifiers for app and service identification, respectively. First,

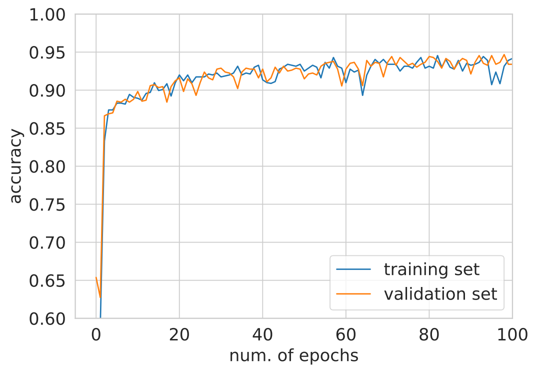
Algorithm	Accuracy %	Precision	Recall	F-Score	# Parameters	Acc Async %	Diff %
Linear SVM	90.80	0.908	0.908	0.907	19843	79.61	-11.2
Logistic Regressor	90.42	0.904	0.904	0.904	243	81.11	-9.3
Nearest Neighbours	92.76	0.925	0.925	0.925	19840	83.45	-9.3
Random Forest	91.57	0.915	0.915	0.915	22320	84.25	-7.3
Gaussian Processes	93.21	0.932	0.928	0.929	73360	82.51	-10.7
Neural Networks							
MLP	94.31	0.943	0.939	0.942	18819	93.38	-0.9
RNN	98.21	0.981	0.982	0.981	391503	95.38	-2.8
CNN	98.87	0.986	0.988	0.988	24963	95.40	-3.5

Table 7.3: Classifiers comparison for the service identification task.

we focus on synchronous sessions results. In general, better performance is achieved through deep NNs (+13.8% on app identification, +8.7% on service classification). Moreover, we observe a significant performance gap between the service and app classification tasks, due to the higher number of classes of the latter: the performance gap is higher than 8% for the benchmark classifiers and ranges from 2 to 6% for NNs. Furthermore, RNN and CNN architectures achieve an accuracy of about 99% for the service identification task (O1) and higher than 95% for the app identification task (O2). The algorithm based on Gaussian Processes performs the best among the benchmark classifiers. In general, the higher the complexity (i.e., the number of parameters, and also hidden layers for NNs), the higher the performance. The only exception to this is provided by CNNs, which present the highest accuracy but use a small number of parameters. This fact confirms the high efficiency of convolutions in processing high amount of data with complex temporal structure, and the effectiveness of parameter sharing. CNNs require only 6% of the variables used up by RNNs, achieving a better accuracy. This also translates into a faster training: in Fig. 7.9, we show the accuracy as a function of the number of epochs for training and validation sets for RNNs and CNNs. The number of epochs required by the CNNs to reach an accuracy higher than 90% is fewer than 20 (Fig. 7.9b), whereas for RNNs convergence is achieved only after 30 epochs (Fig. 7.9a).



(a) RNN



(b) CNN

Figure 7.9: Accuracy *vs* number of epochs for training and validation sets for the app identification task.

A deeper look at the performance of CNNs is provided by the confusion matrices of Fig. 7.8, whose rows and columns respectively represent true and predicted labels, and all values are normalized between 0 and 1. For the service classification task (Fig. 7.8a), CNNs only misclassify the video streaming sessions: 2% of those are labeled as video calls. For the app identification task (Fig. 7.8b), errors (4%) mainly occur for Skype and WhatsApp videocalls. These errors are understandable, as these are both interactive real-time video applications and, as such, their traffic patterns bear similarities. The lowest performance is found for Vimeo traces, for which 88% of the sessions are correctly classified. Here our CNN-based classifier confuses them with the other video applications for both streaming service (Youtube - 3%) and real-time calling (WhatsApp and Skype - 6% and 3%, repectively).

Asynchronous Sessions Results

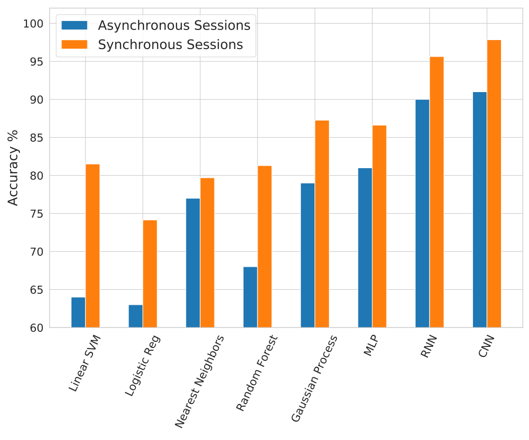
As shown in the last two columns of Tables 7.2 and 7.3, the algorithms' accuracy is affected by asynchronous sessions. As expected, we observe a general decrease in the accuracy for all the algorithms (-6.0% for service identification, -7.7% for app identification, on average). However, for both classification problems, neural network-based approaches are more robust to the asynchronous case, showing a performance degradation of -4.3%, while the degradation of standard algorithms is -8.4%.

Impact of Different Window Sizes

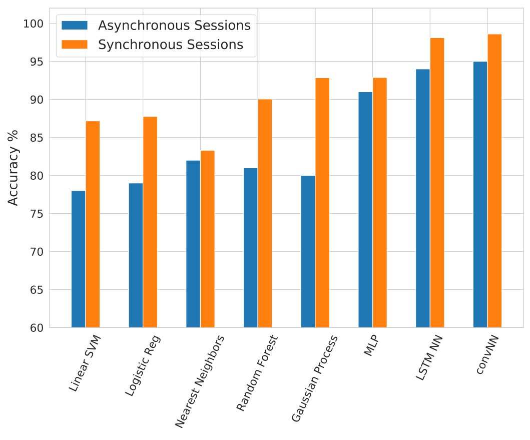
Fig. 7.11 shows the classification accuracy as a function of the window size, W. For the app identification task, 40 seconds suffice for CNNs and RNNs to reach accuracies higher than 90%, with negligible additional improvements for longer observation periods. Periods shorter than 40 seconds provide less accurate results. Similar trends are observed for the service classification task. However, in this case after 20 seconds the accuracy of CNNs and RNNs is already higher than 90%, due to the smaller number of classes. In summary, the ability of CNNs and RNNs to extract representative statistical features from a session grows with the input data length. In our tasks, deep NN algorithms become very effective as monitoring intervals get longer than 20 seconds.

7.1.10 Unsupervised Traffic Profiling

Next, we analyze the mobile traffic exchanged within the four selected cell sites. The traffic load is modeled in terms of aggregated traffic dynamics and type of service requests over the 24 hours of a day. The identification of mobile traffic, for each of the considered services, is performed using the trained classifiers with the unlabeled dataset. Formally, for each eNodeB, the corresponding unlabeled dataset is stored into matrix $\boldsymbol{X}'$, whose size is $M' \times N$, where M' corresponds to the number of monitored RNTIs (sessions) and N to the number of collected samples per session.

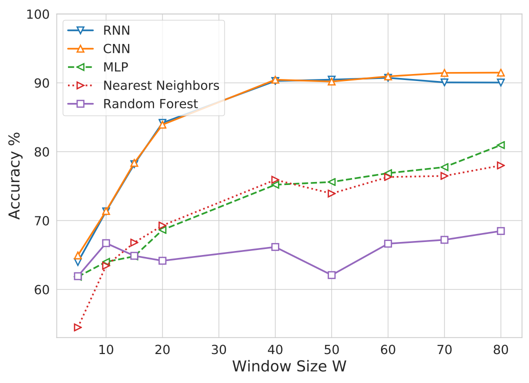


(a) App identification.

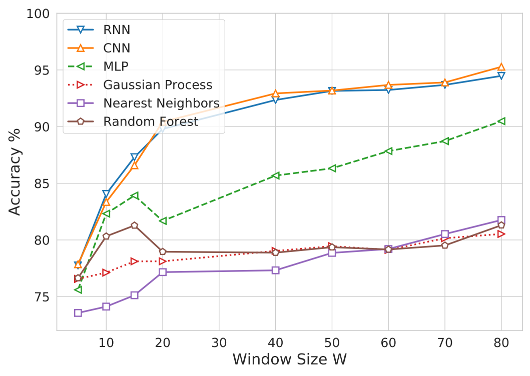


(b) Service identification.

Figure 7.10: Effect of Synchronization vs Asynchronization procedures



(a) App identification.



(b) Service identification.

Figure 7.11: Accuracy *vs* window lengths for app and service classification tasks.

Given $\boldsymbol{X}'$, as input, the classifier c computes the output $\boldsymbol{Y}'$, whose analysis provides a detailed characterization of the mobile user requests for the eNodeB within the monitored time span. Vectors $\boldsymbol{x}'_m$ and $\boldsymbol{y}'_m = c(\boldsymbol{x}'_m)$ respectively indicate the m-th row of matrices $\boldsymbol{X}'$ and $\boldsymbol{Y}'$. In this part, we restrict our attention to the classification of services, and use the CNN classifier, as it yields the highest accuracy.

Aggregated Traffic Analysis

Fig. 7.12 shows the *aggregated* traffic demand for the four selected eNodeBs over the 24 hours of a typical day, where each curve has been normalized with respect to the maximum traffic peak occurred during the day for the corresponding eNodeB.

The four traffic profiles have a different trend, which depends on the characteristics of the served area (demographics, predominant land use, etc.), as confirmed by [150]. **PobleSec** is a residential neighborhood and, as such, presents traffic peaks during the evening, at 5 and 11 pm. **Born** is instead a downtown district with a

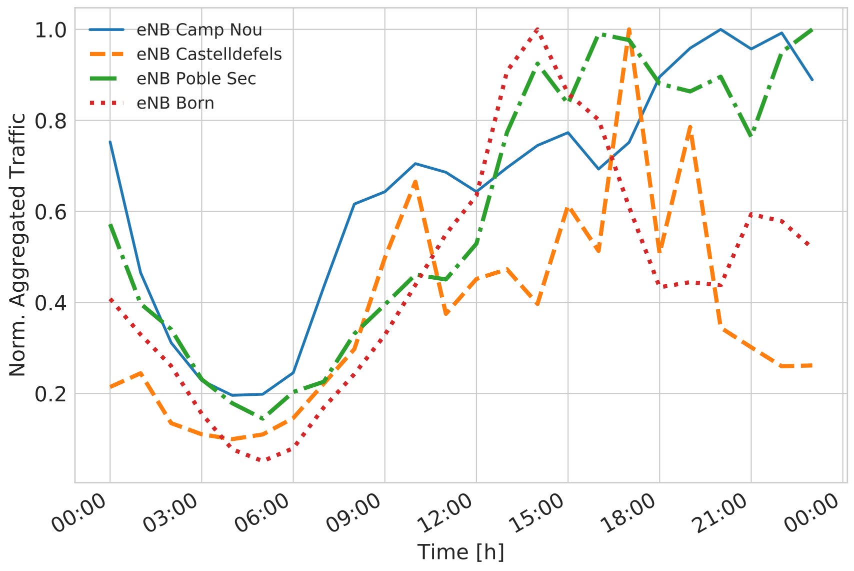


Figure 7.12: Daily Aggregated Traffic for the four eNodeBs.

mixed residential, transport and leisure land use. Two peaks are detected: the highest is at lunchtime around 2 pm, whereas the second one is at dinner time, from 9 pm. This traffic behavior is likely due to the many restaurants and bars in the area. **CampNou** is mainly residential and presents a similar profile to PobleSec. However, Barcelona FC stadium is located in this area, and three football matches took place during the monitored period (events started at 8:45 pm and ended at 10:45 pm). As expected, a higher traffic intensity is observed during the football match hours. In particular, we registered a high amount of traffic exchanged between 7 pm to 1 am, i.e., before, during and after the events. This behavior is probably due to the movement of people attending the matches. **Castelldefels** is a suburban and low populated area with a university campus. The traffic variation suggests a typical office profile with traffic peaks at 10 am and 5 pm. However, in this radio cell the amount of traffic exchanged is the lowest observed across all eNodeB sites, i.e., 6.8 Gb/hour in the peak hours. The highest traffic intensity was measured in Camp Nou, reaching a peak of 106.1 Gb/hour (29.5 Mb/s on average). Intermediate peak values are detected in Poble Sec and Born, amounting to 49.7 Mb/s and 46.1 Mb/s, respectively. The only common pattern among the four areas is the low traffic intensity at night, approximately between 2 am and 7 am.

Traffic Decomposition at Service Level

The set of applications that we have labeled is limited to those apps and services that dominate the radio resource usage. However, additional apps may also be present in the monitored traffic, such as Facebook, Instagram, Snapchat, etc. These apps generate mixed content, including audio-streaming, video-streaming and video-calling. Additional service types may also be generated by, e.g., web-browsing and file downloading. While in the present work the classifiers were not trained to specifically track these apps, for a robust classification outcome, it is desirable that the audio and video streams that they generate will either be captured and classified into the correct service class, or at least flagged as unknown. To locate those traffic patterns for which our classifier may produce inaccurate results, in our analysis we account for the detection of out-of-distribution (OOD) sessions, i.e., of DCI traces that show different traffic dynamics from those learned at training time. To identify these "statistical outliers", the DCI data from each new session, $\boldsymbol{x}'_m$, is fed to the CNN and the corresponding softmax output vector $\boldsymbol{y}'_m = [y'_{m1}, \dots, y'_{mK}]^T$ is used to discriminate whether $\boldsymbol{x}'_m$ is OOD or not, following the rationale in [151][152]. In detail, the k-th softmax output corresponds to the probability estimate that a given input session $\boldsymbol{x}'_m$ belongs to class k, i.e., $y'_{mk} = \text{Prob}(\boldsymbol{x}'_m \in \text{class } k)$, with $k = 1, \dots, K$. The classifier chooses the class $k^\star$ that maximizes this probability, i.e.,

$$k^\star = \operatorname*{argmax}_k y'_{mk}. \tag{7.5}$$

If a new app, not considered in the training phase, generates sessions having similar characteristics to those in the training set, namely, audio-streaming, video-streaming or real time video-calls, we expect the CNN to generalize well and return similar vectors at the output of the softmax layer. That is, the softmax vector that is outputted at runtime for the new app should be sufficiently "close" to the output learned by the classifier from the labeled dataset, as the new signal bears statistical similarities with those learned in the training phase. In this case, it makes sense to accept the session and classify it as belonging to class $k^\star$. Otherwise, the session would be classified as OOD.

The problem at hand, boils down to assessing whether the softmax output $\boldsymbol{y}'_m$ belongs to the statistical distribution learned by the CNN or it is an outlier. This amounts to performing outlier detection in a multivariate setting, with $\boldsymbol{y}'_m \in [0,1]^K$, $\sum_k y'_{mk} = 1$. Among the many algorithms that can be used to this purpose, we adopt the method based on Kernelized Spatial Depth (KSD) functions of [153] as it is lightweight and does not require the direct estimation of the probability density function (pdf) of the softmax output layer, which is a critical point, as good estimates require training over many points. Briefly, for a vector $\boldsymbol{y} \in \mathbb{R}^K$, we define the spatial sign function as $S(\boldsymbol{y}) = \boldsymbol{y}/\|\boldsymbol{y}\|$ if $\boldsymbol{y} \neq \boldsymbol{0}$ and $S(\boldsymbol{y}) = \boldsymbol{0}$ if $\boldsymbol{y} = \boldsymbol{0}$, where $\|\boldsymbol{y}\| = (\boldsymbol{y}^T\boldsymbol{y})^{1/2}$ is the norm-2. If $\mathcal{Y}_k$ is a training set containing ℓ softmax output vectors for a certain class k, $\mathcal{Y}_k = \{\boldsymbol{y}_1^{(k)}, \boldsymbol{y}_2^{(k)}, \dots, \boldsymbol{y}_\ell^{(k)}\}$, the *sample spatial depth* associated with a new softmax output vector $\boldsymbol{y}'_m$ is:

$$D(\boldsymbol{y}'_m, \mathcal{Y}_k) = 1 - \frac{1}{|\mathcal{Y}_k \cup \boldsymbol{y}'_m| - 1} \left\| \sum_{\boldsymbol{y} \in \mathcal{Y}_k} S(\boldsymbol{y} - \boldsymbol{y}'_m) \right\|. \tag{7.6}$$

Note that $D(\boldsymbol{y}'_m, \mathcal{Y}_k) \in [0,1]$ provides a *measure of centrality* of the new point $\boldsymbol{y}'_m$ with respect to the points in the training set $\mathcal{Y}_k$. In particular, if $D(\boldsymbol{y}'_m, \mathcal{Y}_k) = 1$, it follows that $\| \sum_{\boldsymbol{y} \in \mathcal{Y}_k} S(\boldsymbol{y} - \boldsymbol{y}'_m) \| = 0$ and the new point is said to be the *spatial median* of set $\mathcal{Y}_k$, i.e., it can be thought of as the "center of mass" of this set. Hence, the spatial depth attains the highest value of 1 at the spatial median and decreases to zero as $\boldsymbol{y}'_m$ moves away from it. The spatial depth can thus be used as a measure of "extremeness" of a new data point with respect to a set. In [153], the spatial depth of Eq. (7.6) is *kernelized*, which means that distances are evaluated using a positive definite kernel map. A common choice, that we also use in our case, is the generalized Gaussian kernel $\kappa(\boldsymbol{x}, \boldsymbol{y})$,

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp(-(\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{y})), \tag{7.7}$$

which provides a measure of similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$. Noting that the square norm can be expressed as

$$\|\boldsymbol{x} - \boldsymbol{y}\|^2 = \boldsymbol{x}^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{x} - 2\boldsymbol{x}^T \boldsymbol{y}, \tag{7.8}$$

kernelizing the sample spatial depth amounts to expanding (7.6) and replacing the inner products with the kernel function κ. This returns the *sample KSD function* (Eq. (4) in [153]).

Session classification procedure in an unsupervised setting: the CNN classifier is augmented through the detection of OOD sessions, as follows:

- *Initialization:* for each class $k = 1, \dots, K$ in the service/app identification task a number of softmax output vectors is computed by the *trained* CNN using the sessions in the training set. These softmax vectors are stored in the set $\mathcal{Y}_k$. Note that, being the results of a supervised training of the CNN, we know that the vectors in $\mathcal{Y}_k$ are all generated by a distribution that is correctly tracked during the supervised learning phase.
- *Feature extraction through the pre-trained CNN:* at runtime, as a new DCI vector $\boldsymbol{x}'_m$ is measured, it is inputted into the pre-trained CNN, obtaining the corresponding softmax output $\boldsymbol{y}'_m$.
- *Classification and OOD detection:* vector $\boldsymbol{y}'_m$ is used with Eq. (7.5) to assess the most probable class $k^\star$. At this point, Algorithm 1 of [153] is utilized to assess whether $\boldsymbol{y}'_m$ is an outlier. In case the vector is classified as an outlier, it is assigned to the OOD class, otherwise it is assigned to class $k^\star$.

Some final remarks are in order. The outlier detection algorithm uses a threshold $t \in [0, 1]$, which allows exploring the tradeoff between false alarm rate and detection rate. Instead, the covariance matrix $\boldsymbol{\Sigma}$ controls the decision boundary for rejecting vectors, driving the tradeoff between the local and global behavior of KSD. If properly chosen, the contours of KSD should closely follow those of the underlying statistical distribution. $\mathbf{\Sigma}$ is learned, for each class k, from the training vectors in $\mathcal{Y}_k$, and for the following results we picked $\mathbf{\Sigma} = \mathbf{\Sigma}_2$ in [153].

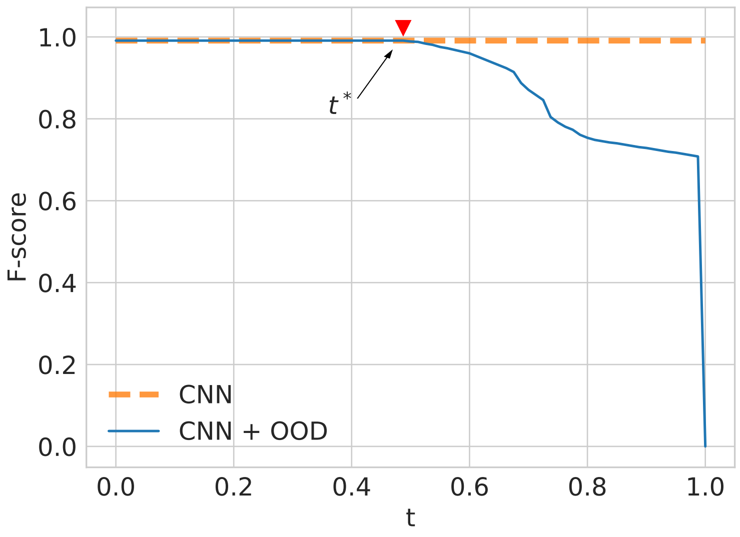
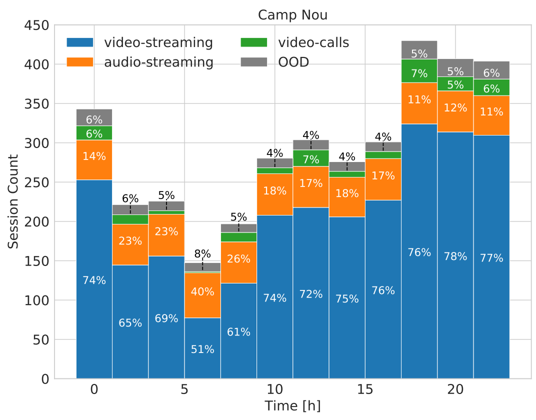
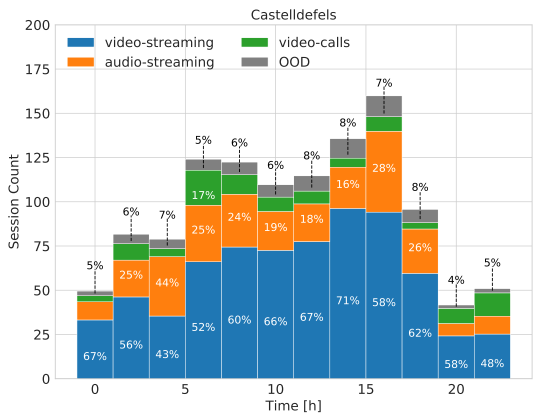


Figure 7.13: Finding threshold $t^\star$ using the CNN with (solid line) and without (dashed line) the OOD detection mechanism.

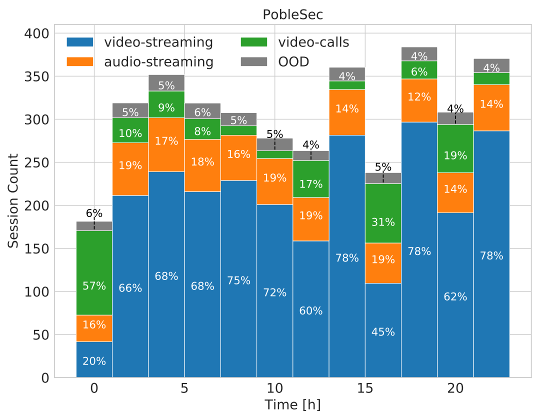
Tuning the OOD threshold t: for each class k, $\mathcal{S}_k$ is obtained from the training dataset. We recall that $\mathcal{S}_k$ is used to compute the covariance matrix associated with the adopted Gaussian kernel, which models the contours of the pdf of the output softmax vectors. The threshold $t \in [0, 1]$ is instead used by the outlier detection algorithm to gauge the (kernelized) distance between the center of mass of set $\mathcal{S}_k$ and a new softmax vector, acquired at runtime. If $t = 1$, the kernelized spatial depth of the new point will always be smaller than or equal to t and all points will be rejected (marked as outliers). This is of course of no use. However, as we decrease t towards 0, we see that more and more points will be accepted, until, for $t = 0$, no rejections will occur. So, t determines the selectivity of the outlier detection mechanism, the higher t, the more selective the algorithm is. For our numerical evaluation, once the sets $\mathcal{S}_k$ are obtained for all classes k, we set this threshold by picking the highest value of it, $t^\star$, for which all the softmax vectors belonging to the test set are accepted, i.e., none of them is marked as an outlier (OOD). In other words, this is equivalent to making sure that the F-Score obtained over the test set from our trained CNN without the OOD mechanism enabled equals that of the CNN classifier augmented with the OOD detection capability. As $t^\star$ is the highest value of t for which all the data in the test set are correctly classified as valid, our approach amounts to tuning the threshold in such a way that the outlier detection algorithm will be as selective as possible, while correctly treating all the data in the test set. In Fig. 7.13, we show the F-Score as a function of t for the CNN algorithm with and without OOD detection. Threshold $t^\star = 0.48$ corresponds to the highest value of t for which the F-Score remains at its maximum, i.e., at the end of the flat region.



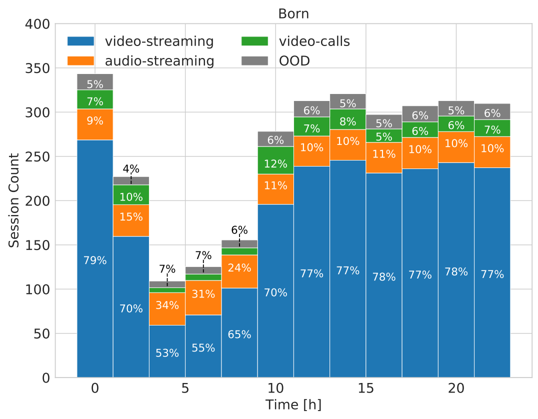
(a) Camp Nou eNodeB.



(b) Castelldefels eNodeB.



(c) Poble Sec eNodeB.



(d) Born eNodeB.

Figure 7.14: Traffic decomposition at service level for the four monitored eNodeBs during the 24 hours of a day.

Experimental analysis of eNodeB traffic: in Fig. 7.14, the traffic decomposition into the considered service classes is shown for the four selected eNodeBs using $t^\star = 0.48$. The percentage of sessions identified as OOD, for which the classifier is uncertain, is also reported at the top of each bar. Common characteristics are observed in all the considered deployments:

- the most used service is video-streaming, with typical shares ranging from 50% to 80%. This confirms the measurements in [138] and [2].
- The least used service is video-call, whose share is typically between 5% and 10%, whereas audio-streaming takes 21% of the total traffic load.
- OOD sessions are consistently well below 8%. Note that this share accounts for all those apps that are not tracked by our classifier, such as texting, web browsing, and file transfers.

Through the proposed service identification approach, we can accurately characterize, at runtime, the used services. Moreover, the traffic decomposition at service level allows one to make some interesting considerations on the land use. For example, in a typical residential area (PobleSec) the audio-streaming service is the one used the least across the four monitored sites, with an average of 16.4%. Instead, in a typical office and university neighborhood (Castelldefels), audio-streaming has the highest traffic share across all sites (22% on average). Born and CampNou, which are two leisure districts, present a similar traffic distribution across the day. We finally remark that, while the traffic profiling results are shown using a time granularity of one hour, our classification tool allows for traffic decomposition at much shorter timescales, i.e., on a per-session basis.

7.2 Application to Network Optimization Frameworks

In this section, we present an application of the LSTM traffic prediction and of the classification model presented in the sections above, applied to the context of energy saving devices. Authors of [25], introduce a novel proactive wake-up scheduler for energy efficiency in 5G. The contribution in this work is restricted to the traffic profiles and to predictive model that serves the wake-up scheduler to take online decisions.

In particular, the proposed approach is based on the discontinuous reception (DRX), which as specified by 3GPP, is the *de facto* power saving mechanism for long-term evolution (LTE) based fourth generation (4G) systems [154, 155] and NR based 5G systems [156]. It has been shown in [157] that the time period that a DRX-enabled mobile device spends monitoring the PDCCH without any data allocation has a major impact on its battery lifetime.

For these reasons, a novel concept is introduced, namely *wake-up scheduling*, to further reduce the power consumption of the mobile device. Proactively knowing the packet arrival times for a forecast horizon, allows the UE to remain at OFF mode for longer periods. The main idea is of using a fixed WuS configuration and then adjusting the scheduling of the wake-up signals dynamically by determining whether to wake-up the device or not, based on traffic predictions over a forecast horizon and a maximum tolerable delay. In addition, differently to previous works in [158, 159, 160], the proposed scheduler is not tied to specific traffic models.

7.2.1 Proposed Wake-up Scheduler

Shortly, we describe the idea behind the wake-up scheduler, which is depicted in the block diagram of Fig. 7.15. The wake-up scheduler does not send a wake-up indicator (WI) as soon as there is a packet in the w-cycle, but waits until some condition is met; for instance, until the number of buffered packets at gNB for a given UE is larger than a predefined buffer size threshold, or until the estimated average buffering delay exceeds a predefined threshold. The main motivation behind not sending WI, as soon as a packet arrives at the gNB but rather waiting and sending the packets consecutively, is that the state-of-the-art modems suffer from large *start-up and power-down stages* [158]. Therefore, it is desired in terms of energy-efficiency that once the modem is at ON mode, it receives multiple packets, and not a single packet. Although, waiting for longer times to buffer packets can eventually increase the buffering delay. This extra buffering delay should not be problematic as long as the average delay is maintained within a maximum bound.

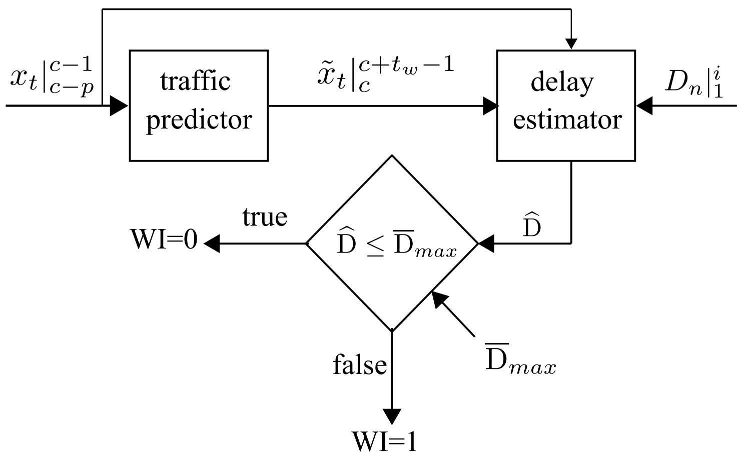

$x_t|_{c-p}^{c-1}$
traffic predictor
$\tilde{x}_t|_c^{c+t_w-1}$
delay estimator
$D_n|_1^i$
$\widehat{\mathrm{D}}$
$\widehat{\mathrm{D}} \leq \overline{\mathrm{D}}_{max}$
true
WI=0
false
WI=1
$\overline{\mathrm{D}}_{max}$

Figure 7.15: Overall block diagram of proactive wake-up scheduler.

For this purpose, the average delay is estimated for k packets, in every w-cycle. In the proposed scheme, *traffic predictor* forecasts the packet arrival times of the target UE for the forecast horizon of one w-cycle based on past packet arrival times. In other words, the traffic predictor observes the session's packet arrival time for p previous TTIs until beginning of the current TTI (c) and then predicts the packet arrival times for the upcoming w-cycle with TTI indexes of $[c, c + t_w)$.

Furthermore, in every w-cycle, a *delay estimator* block estimates the average buffering delay ($\widehat{\mathrm{D}}$) of k packets. If $\widehat{\mathrm{D}}$ is higher than a maximum tolerable delay $\overline{\mathrm{D}}_{max}$, the network realizes that the only way to have shorter delay is by sending WI=1 promptly. Otherwise ($\widehat{\mathrm{D}}<\overline{\mathrm{D}}_{max}$), it leaves the UE to remain in OFF mode for at least another w-cycle. Finally, a *delay comparator* block performs the task of comparison and decision making (i.e., whether to send WI=1 or WI=0) accordingly.

7.2.2 Traffic Predictor and Dataset

The traffic prediction can be formulated as a time series forecasting problem, where the packet arrivals at each TTI are defined as the values of the time series. To this end, we tailor the stacked LSTM neural network architecture [161] presented in Chapter 5 to predict the next packet arrivals over a finite horizon. The proposed architecture for the traffic predictor is depicted in Fig. 7.16.

The performance of the proactive wake-up scheduler is investigated using real video and audio streaming traces. For this, we monitored one operative network in Spain during one month using the online watcher presented in [16]. We have selected only those traces gathered during the night hours (1am - 6am) to be sure that the selected cell is serving very few users. This allows us to assume that our traces are not affected by the packet scheduler at the base station, since an adequate number of radio resources per TTI is available to accommodate all the transmitting UEs. Our dataset includes two columns: the Identifier of the UE, and the timestamp of the packet arrival (with TTI granularity). The classifier introduced in 7.1 is used to properly select the traces of the apps of interest. The collected dataset consists of 1500 sessions of different traffic type. For the sake of comparison, we also generated Poisson traffic with mean packet arrival rate of 0.2 p/TTI (video and audio traffics have varying packet arrival rates up to 0.2 p/TTI) and added them to the dataset.

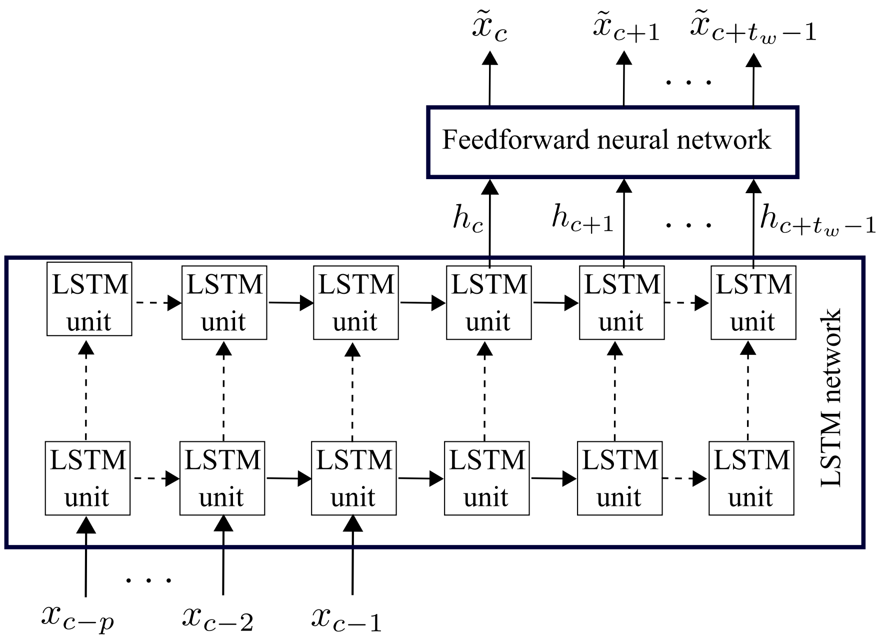


Figure 7.16: Proposed architecture for the packet arrival time prediction.

The dataset with size z for a particular traffic type is represented by $x_t|_1^z$, where x_t indicates the packet arrival time during the t^{th} TTI. As shown in Fig. 7.15 and 7.16, the proposed network observes $x_t|_{c-p}^{c-1}$ and, then, predicts the traffic in the upcoming w-cycle $\tilde{x}_t|_c^{c+t_w-1}$ by delaying the prediction for the duration of t_w. Finally, the output of the LSTM network ($h_t|_c^{c+t_w-1}$) is fed to a fully connected neural network that performs the actual prediction. The first layer size corresponds to p observed TTIs, while the last layer output has a length equal to future horizon t_w.

The implementation of the traffic prediction algorithm was performed in Python, using Keras and Tensorflow, as backend. The chosen hyperparameters are reported in Table 7.4. The number of hidden layers is fixed to 5, which is the number giving a good trade-off between prediction accuracy and model complexity. For the training part, we used the Adam's algorithm [116] as optimizer and the Mean Absolute

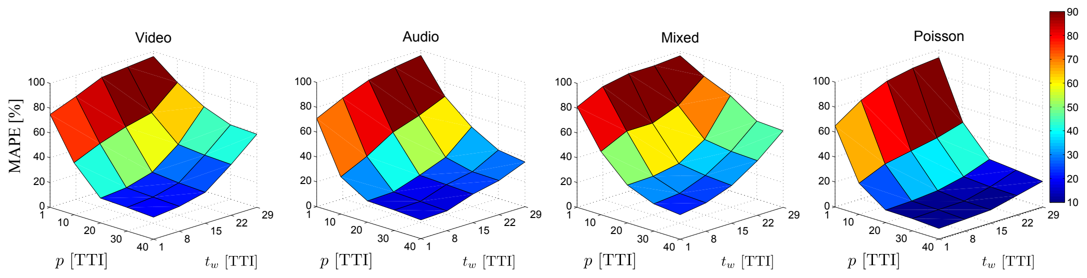


Figure 7.17: MAPE as function of number of past observations p and forecast horizon t_w for different traffic types.

Percentage Error (MAPE) as loss function. We define the MAPE as follows,

$$\text{MAPE} = \frac{100\%}{t_w} \sum_{t=c}^{c+t_w-1} \frac{|\tilde{x}_t - x_t|}{x_t} \tag{7.9}$$

where $\tilde{x}_t$ is the predicted packet arrival time on the t^{th} TTI.

Table 7.4: Training hyperparameters

Initial learning rate	0.001
Number of epochs	100
Number of LSTM hidden states	64
Number of LSTM hidden layers	5
Number of feed-forward hidden layers	1
Optimization algorithm	Adam
Loss function	MAPE

7.2.3 Numerical Results

Numerical results are given to evaluate the proposed proactive schemes in terms of prediction accuracy, packet delay and power consumptions and energy saving. However, we are interested to show the results related to the prediction model and the delay of the different traffic types. For all results, the reader is invited to refer to [25].

The impact of the length of the horizon (t_w) and the number of previous observations (p) on the prediction errors is illustrated in Fig. 7.17. For shorter w-cycles, the predictions follow the actual values closely, whereas for larger w-cycles, the prediction error is bigger: longer forecast horizons (t_w) decrease the accuracy of the predictor, as expected. Furthermore, as it can be observed, the MAPE reduces with a larger number of observations (p) for all four traffic types. Also, the accuracy is smaller for Poisson packet arrivals than for video and audio traffics, due to its simpler traffic pattern. For Poisson traffic, the MAPE increases around 15% when t_w increases from 10 to 30 TTIs for given $p = 20$ TTIs.

As shown in Fig. 7.17, it is desirable to reduce t_w and enlarge p. However, in terms of power consumption, such a reduction of the w-cycle would contribute to a higher energy consumption due to frequent checking of wake-up signaling. Additionally, a higher number of past observations p involves a longer memory length of the LSTM network and a large amount of information that must be stored for a precise traffic prediction. As a result, the floating point operations per second (FLOPS) of the LSTM network increases. This complexity overhead can become very high, especially if the number of users per cell increases.

Fig. 7.18 shows the empirical cumulative distribution function (CDF) of packet delay for the four different traffic types. Two different sets of performance results, in terms of power consumption and delay, are presented. Namely, (1) wake-up scheme without scheduler (**WuS**) that is considered as a benchmark scheme, and (2) wake-up scheme with proactive scheduler (**Pro**).

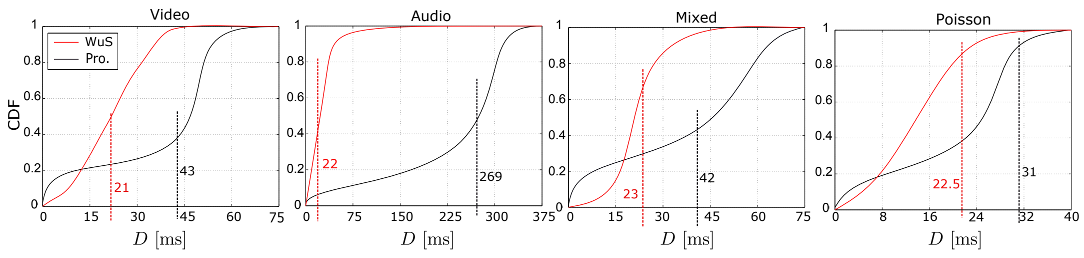


Figure 7.18: The CDF graphs of buffering delay of packets for WuS and proactive scheduler under different traffic types. The dashed lines and corresponding numbers represent average delays caused by the particular method.

Generally, video streaming sessions are much longer than that of the audio traffic, and packets arrive burstly (implying high self-similarity). As it can be seen for video results of proactive scheduler, a large number of packets are served with near to zero delay, and the reason is due to the consecutive packet arrivals that are served while the inactivity timer is triggered. At the same time, a large number of packets are served with delays larger than the maximum delay budget of video (40 ms), and this comes from the fact that the proactive scheduler is a greedy method and waits until the average buffering delay approaches to $\overline{\mathrm{D}}_{max}$. As compared to the proactive scheduler, WuS has a lower and consistent delay regardless of the traffic types.

7.2.4 Conclusions

In the presented work, the traffic prediction model based on LSTM network and data collected from LTE PDCCH are used to propose a novel proactive wake-up scheduler. The feasibility of proactive scheduler based on user traffic prediction has been investigated and the simulation results using different traffic types show that proactive scheduler has lower energy consumption than the wake-up scheme without scheduler. Moreover, the promising results motivate jointly considering user traffic prediction and wake-up scheduler in order to reduce the energy consumption of users under different traffic circumstances.

Chapter 8

Conclusions

8.1 Summary of Achievements

The challenges that arise from the next generation mobile networks encompass a multitude of elements, that include the increase of requirements to support a more complex infrastructure. In this thesis, we leverage the potential and the capabilities of ML algorithms to tackle some essential features in the context of mobile traffic analysis. The thesis is organized into one preliminary part and four technical parts:

- in Chapter 1, 2, and 3, we introduce the context and the problem. Moreover, we present related works on the analysis of traffic datasets based on both statistical and learning methods; also, in Chapter 3, we provide the reader of a brief background regarding the ML framework that is adopted throughout the thesis;
- in Chapter 4, we describe the data collection system and the measurement campaign;
- in Chapter 5, we present the analysis of the collected traffic traces and we derive a model for traffic prediction;
- in Chapter 6, we define a methodology for the detection of urban anomalies using the collected data; we provide two different use-cases;
- in Chapter 7, we perform a classification of the applications using solely the physical data that we collect; moreover, we describe an application for the classifier and for the traffic predictor is given, in the context of energy-saving devices.

Next, we briefly report the summary of contributions for each specific technical part of the thesis.

8.1.1 Data Analysis using LTE PDCCH Information

In Chapter 4, we described the measurements and the dataset that have been used to conduct the analysis throughout the rest of the thesis. The analysis has shown that using a simple and inexpensive LTE sniffer, we were able to acquire a millisecond-resolution dataset that allows to perform extensive study on the user mobile traffic for a given LTE eNodeB. This step is fundamental in any ML pipeline development, since the quality of the collected data can affect the subsequent analysis and results. Moreover, there is no need for the intervention of a mobile network operator, and the user's privacy is preserved since we are decoding only the scheduling information of the mobile users exchanged on the LTE PDCCH.

In Chapter 5, we performed an analysis to describe the recurrent patterns in the mobile traffic. We showed that the aggregate traffic can be modeled with a simple discrete-time Markov chain, and, through unsupervised learning methods, we showed differences and similarities between different traffic types. Moreover, we proved that using LSTM neural networks, we can implement a multi-steps traffic predictor that beat the performance of wide-used statistical methods like ARIMA: this can be valuable also in related works, where researchers, for example, need to generate traffic patterns in their experiments.

8.1.2 Urban Anomaly Detection with Mobile Traffic

Chapter 6 has been devoted to describe the detection of urban anomalies using solely the collected dataset. Such topic is relevant in the context of integrating the mobile network with other urban infrastructures (e.g. smart cities), since an early detection can prevent events that may be harmful to public safety. In this part of the work, we proved that urban anomalies that derive from crowd-gathering events due, for example, to sport events, can be identified through the analysis of LTE data. First, we considered a simple case, where the event is supervisedly marked; then, we extended our method to multiple type of events and we performed the anomaly detection using a semi-supervised framework, that includes the utilization of autoencoders structures.

8.1.3 Additional Applications: Traffic Identification and Prediction

In Chapter 7, we dealt with two applications that can be exploited by the network management to optimize the network performances. First, we investigated the possibility of identifying which app has been run on the users' device. We solved this problem using a supervised methodology, in which we monitored a mobile phone that run the most wide-used applications in three categories: video-streaming, audio-streaming, and real-time video-calling applications. An extensive benchmark for the identification for the app and for the services has been presented: the results showed that adapting deep-learning structures with CNNs, we can obtain the upmost results with more than 90% accuracy. Finally, we reported a sample application of the LSTM traffic prediction tool that we presented in Chapter 5. In this case, the main author integrated the traffic prediction in a wake-up scheme, that allows to save energy in a 5G device using the DRX mechanism.

8.2 Future Works

In this thesis, we have implemented solutions for a limited part of the problems from studying mobile network user traffic. However, due to the myriads of aspects to be considered in this topic, there are open issues that can be relevant in order to define additional future works.

8.2.1 Extensive Dataset Collection

We focus the collection of data from LTE base stations for which we know the correspondent land-uses. This allowed us to infer a characterization on the users' traffic profiles and on their temporal dynamics. However, a more extended dataset would permit the description of different scenarios, and may have included cases that we were not able to define. Also, a dataset that cover larger urban areas (like the one released by mobile network operators), would allow to describe the geographical correlation of the traffic and, possibly, to study the users' mobility and, for example, to analyze the effects and problems of intercell roaming with regard to the network performances.

8.2.2 Network Management Optimization

The application of the thesis' results in existing and/or developing frameworks should be evaluated in order to demonstrate their effectiveness. In Chapter 7, we reported the utilization of the implemented prediction model that has been used in a wake-up scheme algorithm for devices with DRX. This is an example in the context of energy-saving devices. The implementation of optimization solutions for the network management should cover different type of applications: for example, since we collected scheduling information, we may develop an optimized scheduling algorithm to allocate the physical resources among the connected users. In these solutions, where the objective is to learn a policy, we should properly refer to RL algorithms, instead of referring only to supervised learning methods.

8.2.3 Integration with Big Data platforms

The analysis has been conducted on data retrieved from the mobile network and the study has been done through an offline approach: this limits the presented results to restricted solutions, in which time and operational complexity are not consistently taken into account. To be formally feasible in a realistic case, the study should consider the effect of elaborating the data streaming in real-time at large-scale: Big data frameworks are functional to this end, and may be included in the evaluation and presented as a trade-off with the offline approach.

Bibliography

[1] NGMN Alliance. "5G white paper". In: *Next generation mobile networks, white paper* (2015).

[2] Cisco. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*. 2017. URL: `www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`.

[3] Jessica Moysen and Lorenza Giupponi. "From 4G to 5G: Self-organized network management meets machine learning". In: *Computer Communications* 129 (2018), pp. 248–268.

[4] ETSI. "Self-Configuration of Network Elements Integration Reference Point (IRP); Common Object Request Broker Architecture (CORBA) Solution Set (SS)(Release 8)". In: *3GPP TR* 32.503 (2008).

[5] Multazamah Alias, Navrati Saxena, and Abhishek Roy. "Efficient cell outage detection in 5G HetNets using hidden Markov model". In: *IEEE Communications Letters* 20.3 (2016), pp. 562–565.

[6] Lorenza Giupponi et al. "A novel approach for joint radio resource management based on fuzzy neural methodology". In: *IEEE Transactions on Vehicular Technology* 57.3 (2008), pp. 1789–1805.

[7] Emil J Khatib et al. "Diagnosis based on genetic fuzzy algorithms for LTE self-healing". In: *IEEE Transactions on vehicular technology* 65.3 (2015), pp. 1639–1651.

[8] Paulo Valente Klaine et al. "A survey of machine learning techniques applied to self-organizing cellular networks". In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2392–2431.

[9] Qingchen Zhang et al. "A survey on deep learning for big data". In: *Information Fusion* 42 (2018), pp. 146–157.

[10] Kritika Walia. "Self Driving Car Market Report| Industry Share, Trends & Growth Analysis To 2025". In: (2019).

[11] Norman Jouppi et al. "Motivation for and evaluation of the first tensor processing unit". In: *IEEE Micro* 38.3 (2018), pp. 10–19.

[12] Li Deng. "The MNIST database of handwritten digit images for machine learning research [best of the web]". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[13] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[14] Vincent D Blondel et al. "Data for development: the d4d challenge on mobile phone data". In: *arXiv preprint arXiv:1210.0137* (2012).

[15] OpenSignal.com. *https://opencellid.org/*.

[16] Nicola Bui and Joerg Widmer. "OWL: a reliable online watcher for LTE control channel measurements". In: *ACM All Things Cellular* (2016).

[17] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[18] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[19] Hoang Duy Trinh et al. "Analysis and Modeling of Mobile Traffic Using Real Traces". In: *PIMRC* (2017).

[20] Arcangela Rago et al. "Unveiling Radio Resource Utilization Dynamics of Mobile Traffic through Unsupervised Learning". In: *2019 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE. 2019, pp. 209–214.

[21] Hoang Duy Trinh, Lorenza Giupponi, and Paolo Dini. "Mobile traffic prediction from raw data using LSTM networks". In: *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE. 2018, pp. 1827–1832.

[22] Hoang Duy Trinh, Lorenza Giupponi, and Paolo Dini. "Urban Anomaly Detection by processing Mobile Traffic Traces with LSTM Neural Networks". In: *Proceedings of the 2019 IEEE International Conference on Sensing, Communication and Networking(SECON)* (2019).

[23] Hoang Duy Trinh et al. "Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach". In: *IEEE Access* 7 (2019), pp. 152187–152201.

[24] Hoang Duy Trinh et al. "Mobile Traffic Classification through Physical Channel Fingerprinting: a Deep Learning Approach". In: *arXiv preprint arXiv:1910.11617* (2019).

[25] Soheil Rostami et al. "Proactive Wake-up Scheduler based on Recurrent Neural Networks". In: *arXiv preprint arXiv:1911.22022* (2019).

[26] Soheil Rostami et al. "PWake-up Scheduling for Energy-Efficient Mobile Devices". In: *arXiv preprint arXiv:1911.33102* (2019).

[27] I Chih-Lin et al. “Toward green and soft: a 5G perspective”. In: *IEEE Communications Magazine* 52.2 (2014), pp. 66–73.

[28] Telecom Italia. “Big data challenge 2015”. In: *aris.me/contents/teaching /data-mining-2015 /project/BigDataChallengeData.html* (2015).

[29] Damien Christophe Jacques et al. “" Genesis of millet prices in Senegal: the role of production, markets and their failures”. In: (2014).

[30] Christopher Smith, Afra Mashhadi, and Licia Capra. “Ubiquitous sensing for mapping poverty in developing countries”. In: *Paper submitted to the Orange D4D Challenge* (2013).

[31] Adeline Decuyper et al. “Estimating food consumption and poverty indices with mobile phone data”. In: *arXiv preprint arXiv:1412.2595* (2014).

[32] Diala Naboulsi et al. “Large-scale mobile traffic analysis: a survey”. In: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 124–161.

[33] OpenSignal.com. *https://opensignal.com/reports/*. 2017.

[34] M Zubair Shafiq et al. “Characterizing geospatial dynamics of application usage in a 3G cellular data network”. In: *2012 Proceedings IEEE INFOCOM*. IEEE. 2012, pp. 1341–1349.

[35] M Zubair Shafiq et al. “Large-scale measurement and characterization of cellular machine-to-machine traffic”. In: *IEEE/ACM Transactions on Networking (ToN)* 21.6 (2013), pp. 1960–1973.

[36] Victor Soto and Enrique Frias-Martinez. “Automated land use identification using cell-phone records”. In: *Proceedings of the 3rd ACM international workshop on MobiArch*. ACM. 2011, pp. 17–22.

[37] Blerim Cici et al. “On the decomposition of cell phone activity patterns and their connection with urban ecology”. In: *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM. 2015, pp. 317–326.

[38] Julian Candia et al. “Uncovering individual and collective human dynamics from mobile phone records”. In: *Journal of physics A: mathematical and theoretical* 41.22 (2008), p. 224015.

[39] Carey Williamson et al. “Characterization of CDMA2000 cellular data network traffic”. In: *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN’05) l*. IEEE. 2005, Z000–719.

[40] Utpal Paul et al. “Understanding traffic dynamics in cellular data networks”. In: *IEEE INFOCOM*. IEEE. Shanghai, China, Apr. 2011.

[41] G Peter Zhang. “Time series forecasting using a hybrid ARIMA and neural network model”. In: *Neurocomputing* 50 (2003), pp. 159–175.

[42] Yi Wang, Michalis Faloutsos, and Hui Zang. “On the usage patterns of multimodal communication: Countries and evolution”. In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 3135–3140.

[43] Juan Camilo Cardona, Rade Stanojevic, and Nikolaos Laoutaris. “Collaborative consumption for mobile broadband: A quantitative study”. In: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. ACM. 2014, pp. 307–318.

[44] Yantai Shu et al. “Wireless Traffic Modeling and Prediction Using Seasonal ARIMA Models”. In: E88B (Jan. 2003).

[45] Jing Wang et al. “Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach”. In: *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE. 2017, pp. 1–9.

[46] Wei-Chiang Hong. “Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting”. In: *Neural Computing and Applications* 21.3 (2012), pp. 583–593.

[47] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[48] Huichu Zhang, Yu Zheng, and Yong Yu. “Detecting urban anomalies using multiple Spatio-temporal data sources”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.1 (2018), p. 54.

[49] Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. “Anomaly detection in wireless sensor networks”. In: *IEEE Wireless Communications* 15.4 (2008), pp. 34–40.

[50] Yongguang Zhang, Wenke Lee, and Yi-An Huang. “Intrusion detection techniques for mobile wireless networks”. In: *Wireless Networks* 9.5 (2003), pp. 545–556.

[51] Yin Zhang et al. “Online Identification of Hierarchical Heavy Hitters: Algorithms, Evaluation, and Applications”. In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. IMC ’04. Taormina, Sicily, Italy, 2004, pp. 101–114. ISBN: 1-58113-821-0.

[52] Hossein Saeedi Emadi and Sayyed Majid Mazinani. “A novel anomaly detection algorithm using DBSCAN and SVM in wireless sensor networks”. In: *Wireless Personal Communications* 98.2 (2018), pp. 2025–2035.

[53] Supriya Mishra and Meenu Chawla. “A comparative study of local outlier factor algorithms for outliers detection in data streams”. In: *Emerging Technologies in Data Mining and Information Security*. Springer, 2019, pp. 347–356.

[54] Md Amran Siddiqui et al. "Sequential Feature Explanations for Anomaly Detection". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.1 (2019), p. 1.

[55] Sarah M Erfani et al. "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning". In: *Pattern Recognition* 58 (2016), pp. 121–134.

[56] Alban Siffer et al. "Anomaly detection in streams with extreme value theory". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2017, pp. 1067–1075.

[57] Shilin He et al. "Experience report: system log analysis for anomaly detection". In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE. 2016, pp. 207–218.

[58] Markus Goldstein and Seiichi Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data". In: *PloS one* 11.4 (2016), e0152173.

[59] Bilal Hussain, Qinghe Du, and Pinyi Ren. "Semi-supervised learning based big data-driven anomaly detection in mobile wireless networks". In: *China Communications* 15.4 (2018), pp. 41–57.

[60] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019).

[61] Ilyas Alper Karatepe and Engin Zeydan. "Anomaly detection in cellular network data using big data analytics". In: *European Wireless 2014; 20th European Wireless Conference*. VDE. 2014, pp. 1–5.

[62] Kashif Sultan, Hazrat Ali, and Zhongshan Zhang. "Call detail records driven anomaly detection and traffic prediction in mobile cellular networks". In: *IEEE Access* 6 (2018), pp. 41728–41737.

[63] Alessandro D'Alconzo et al. "A distribution-based approach to anomaly detection and application to 3G mobile traffic". In: *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE. 2009, pp. 1–8.

[64] Lorenzo Fernández Maimó et al. "A self-adaptive deep learning-based system for anomaly detection in 5G networks". In: *IEEE Access* 6 (2018), pp. 7700–7712.

[65] PV Bindu and P Santhi Thilagam. "Mining social networks for anomalies: Methods and challenges". In: *Journal of Network and Computer Applications* 68 (2016), pp. 213–229.

[66] Max Landauer et al. "Time series analysis: unsupervised anomaly detection beyond outlier detection". In: *International Conference on Information Security Practice and Experience*. Springer. 2018, pp. 19–36.

[67] Guangjun Wu et al. "A Fast kNN-Based Approach for Time Sensitive Anomaly Detection over Data Streams". In: *International Conference on Computational Science*. Springer. 2019, pp. 59–74.

[68] Zhaoli Liu et al. "An integrated method for anomaly detection from massive system logs". In: *IEEE Access* 6 (2018), pp. 30602–30611.

[69] Gerhard Münz, Sa Li, and Georg Carle. "Traffic anomaly detection using k-means clustering". In: *GI/ITG Workshop MMBnet*. 2007, pp. 13–14.

[70] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. Vol. 589. John wiley & sons, 2005.

[71] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. "Special issue on learning from imbalanced data sets". In: *ACM Sigkdd Explorations Newsletter* 6.1 (2004), pp. 1–6.

[72] Clifton Phua, Damminda Alahakoon, and Vincent Lee. "Minority report in fraud detection: classification of skewed data". In: *Acm sigkdd explorations newsletter* 6.1 (2004), pp. 50–59.

[73] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[74] Shamoz Shah and Madhu Goyal. "Anomaly Detection in Social Media Using Recurrent Neural Network". In: *International Conference on Computational Science*. Springer. 2019, pp. 74–83.

[75] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. "Deep learning in mobile and wireless networking: A survey". In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2224–2287.

[76] Xi Ouyang et al. "Deepspace: An online deep learning framework for mobile big data to understand human mobility patterns". In: *arXiv preprint arXiv:1610.07009* (2016).

[77] Dario Bega et al. "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning". In: *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE. 2019, pp. 280–288.

[78] Tae-Young Kim and Sung-Bae Cho. "Web traffic anomaly detection using C-LSTM neural networks". In: *Expert Systems with Applications* 106 (2018), pp. 66–76.

[79] Yang Yu, Jun Long, and Zhiping Cai. "Network intrusion detection through stacking dilated convolutional autoencoders". In: *Security and Communication Networks* 2017 (2017).

[80] Yuta Kawachi, Yuma Koizumi, and Noboru Harada. "Complementary set variational autoencoder for supervised anomaly detection". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 2366–2370.

[81] Rosario G Garroppo and Saverio Niccolini. "Anomaly detection mechanisms to find social events using cellular traffic data". In: *Computer Communications* 116 (2018), pp. 240–252.

[82] Dandelion API. *Open Big Data*. `https://dandelion.eu/datamine/open-big-data/`. [Online; accessed 11-June-2019]. 2018.

[83] Giuseppe Aceto et al. "Mobile encrypted traffic classification using deep learning". In: *Network Traffic Measurement and Analysis Conference (TMA)*. IEEE. Vienna, Austria, June 2018.

[84] Chaoyun Zhang, Xi Ouyang, and Paul Patras. "Zipnet-gan: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network". In: *International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM. Seoul-Incheon, South Korea, 2017.

[85] Zhengyang Chen et al. "Automatic mobile application traffic identification by convolutional neural networks". In: *IEEE Trustcom/BigDataSE/ISPA*. IEEE. Tianjin, China, Aug. 2016.

[86] Yeongrak Choi et al. "Automated classifier generation for application-level mobile traffic identification". In: *IEEE Network Operations and Management Symposium (NOMS)*. IEEE. Hawaii, USA, Apr. 2012.

[87] Yanjie Fu et al. "Service usage classification with encrypted internet traffic in mobile messaging apps". In: *IEEE Transactions on Mobile Computing* 15.11 (2016), pp. 2851–2864.

[88] Jie Yang et al. "An empirical investigation into CDMA network traffic classification based on feature selection". In: *International Symposium on Wireless Personal Multimedia Communications (WPMC)*. IEEE. Taipei, Taiwan, Dec. 2012.

[89] Xue Han et al. "Maximum entropy based IP-traffic classification in mobile communication networks". In: *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. Shanghai, China, Apr. 2012.

[90] Yang Liu et al. "A cascade forest approach to application classification of mobile traces". In: *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. Barcelona, Spain, Apr. 2018.

[91] Brendan Saltaformaggio et al. "Eavesdropping on Fine-Grained User Activities Within Smartphone Apps Over Encrypted Network Traffic". In: *USENIX Workshop on Offensive Technologies (WOOT)*. Austin, TX, USA, Aug. 2016.

[92] Vincent F Taylor et al. "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic". In: *IEEE European Symposium on Security and Privacy*. IEEE. Saarbrücken, Germany, Mar. 2016.

[93] Sophon Mongkolluksamee, Vasaka Visoottiviseth, and Kensuke Fukuda. "Enhancing the performance of mobile traffic identification with communication patterns". In: *IEEE 39th Annual Computer Software and Applications Conference (COMPSAC)*. IEEE. Taichung, Taiwan, July 2015.

[94] Giuseppe Aceto et al. "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges". In: *IEEE Transactions on Network and Service Management* 16.2 (2019), pp. 445–458.

[95] Shahbaz Rezaei and Xin Liu. "Deep learning for encrypted traffic classification: An overview". In: *IEEE communications magazine* 57.5 (2019), pp. 76–81.

[96] Pan Wang et al. "A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning". In: *IEEE Access* 7 (2019), pp. 54024–54033.

[97] Fang-Mei Tseng and Gwo-Hshiung Tzeng. "A fuzzy seasonal ARIMA model for forecasting". In: *Fuzzy Sets and Systems* 126.3 (2002), pp. 367–376.

[98] Dagnachew Azene Temesgene, Marco Miozzo, and Paolo Dini. "Dynamic control of functional splits for energy harvesting virtual small cells: a distributed reinforcement learning approach". In: *Computer Communications* 148 (2019), pp. 48–61.

[99] David E Rumelhart and James L McClelland. "On learning the past tenses of English verbs". In: (1986).

[100] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. 2013, pp. 1310–1318.

[101] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[102] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: (1999).

[103] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[104] Matteo Gadaleta and Michele Rossi. "Idnet: Smartphone-based gait recognition with convolutional neural networks". In: *Pattern Recognition* 74 (2018), pp. 25–37.

[105] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[106] KDNuggets.com. *Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis*. 2019. URL: `https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html`.

[107] Thomas R Henderson et al. "Network simulations with the ns-3 simulator". In: *SIGCOMM demonstration* 14.14 (2008), p. 527.

[108] "E-UTRA; Physical layer procedures". In: *3GPP TS, vol. 36.213* (2016).

[109] 3GPP. "5G - Service requirements for next generation new services and markets". In: *3GPP TR, vol. 22.261* (2018).

[110] Ismael Gomez-Miguelez et al. "srsLTE: an open-source platform for LTE evolution and experimentation". In: *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. ACM. 2016, pp. 25–32.

[111] ETSI. "E-UTRA; Physical channel and modulation". In: *3GPP TS* 36.211 (2016), p. V13.

[112] G Auer et al. *EARTH Deliverable D2. 3: Energy efficiency analysis of the reference systems, areas of improvements and target breakdown*. 2013.

[113] Mohammad T Kawser et al. "Downlink SNR to CQI Mapping for Different MultipleAntenna Techniques in LTE". In: *International Journal of Information and Electronics Engineering* 2.5 (2012), p. 757.

[114] Donald F Morrison, Lauriston C Marshall, and Harry L Sahlin. "Multivariate statistical methods". In: (1976).

[115] Giovanna Menardi. "Density-based Silhouette diagnostics for clustering methods". In: *Statistics and Computing* 21.3 (2011), pp. 295–308.

[116] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[117] Francesco Calabrese, Laura Ferrari, and Vincent D Blondel. "Urban sensing using mobile phone network data: a survey of research". In: *Acm computing surveys (csur)* 47.2 (2015), p. 25.

[118] Salil S Kanhere. "Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces". In: *2011 IEEE 12th International Conference on Mobile Data Management*. Vol. 2. IEEE. 2011, pp. 3–6.

[119] Afif Osseiran et al. "Scenarios for 5G mobile and wireless communications: the vision of the METIS project". In: *IEEE Communications Magazine* 52.5 (2014), pp. 26–35.

[120] Zhiheng Huang, Wei Xu, and Kai Yu. "Bidirectional LSTM-CRF models for sequence tagging". In: *arXiv preprint arXiv:1508.01991* (2015).

[121] Donghwoon Kwon et al. "A survey of deep learning-based network anomaly detection". In: *Cluster Computing* (2017), pp. 1–13.

[122] Jason Wang and Luis Perez. "The effectiveness of data augmentation in image classification using deep learning". In: *Convolutional Neural Networks Vis. Recognit* (2017).

[123] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. "Smart augmentation learning an optimal data augmentation strategy". In: *IEEE Access* 5 (2017), pp. 5858–5869.

[124] Tijmen Tieleman and Geoffrey Hinton. “Divide the gradient by a running average of its recent magnitude”. In: *Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[125] Yue Zhao, Zain Nasrullah, and Zheng Li. “PyOD: A Python Toolbox for Scalable Outlier Detection”. In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: `http://jmlr.org/papers/v20/19-011.html`.

[126] John C Platt et al. “Estimating the support of a high-dimensional distribution”. In: *Technical Report MSR-T R-99–87, Microsoft Research (MSR)* (1999).

[127] Hans-Peter Kriegel, Arthur Zimek, et al. “Angle-based outlier detection in high-dimensional data”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2008, pp. 444–452.

[128] M-L SHYU. “A novel anomaly detection scheme based on principal component classifier”. In: *Proc. ICDM Foundation and New Direction of Data Mining workshop, 2003*. 2003, pp. 172–179.

[129] Vaishali Ganganwar. “An overview of classification algorithms for imbalanced datasets”. In: *International Journal of Emerging Technology and Advanced Engineering* 2.4 (2012), pp. 42–47.

[130] *FC Barcelona - Official Website*. 2018. URL: `https://www.fcbarcelona.com`.

[131] *El Rastro — Wikipedia, The Free Encyclopedia*. 2010. URL: `https://en.wikipedia.org/wiki/El_Rastro`.

[132] *July 2018 Lunar Eclipse — Wikipedia, The Free Encyclopedia*. 2010. URL: `https://en.wikipedia.org/wiki/July_2018_lunar_eclipse`.

[133] Sepp Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pp. 107–116.

[134] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[135] Ana Maria Bianco et al. “Outlier detection in regression models with arima errors using robust estimates”. In: *Journal of Forecasting* 20.8 (2001), pp. 565–579.

[136] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. “Efficient algorithms for mining outliers from large data sets”. In: *ACM Sigmod Record*. Vol. 29. 2. ACM. 2000, pp. 427–438.

[137] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation-based anomaly detection”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1 (2012), p. 3.

[138] Ericsson. *Ericsson Mobility Report June 2018*. 2018. URL: `https://www.ericsson.com/en/mobility-report/reports/june-2018`.

[139] Shanzhi Chen and Jian Zhao. "The requirements, challenges, and technologies for 5G of terrestrial mobile telecommunication". In: *IEEE communications magazine* 52.5 (2014), pp. 36–43.

[140] EU EARTH: Energy Aware Radio and neTwork tecHnologies. *D2.3: Energy efficiency analysis of the reference systems, areas of improvements and target breakdown*. Deliverable D2.3, `www.ict-earth.eu`. 2010.

[141] Fengli Xu et al. "Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment". In: *IEEE/ACM Transactions on Networking* 25.2 (2017).

[142] Cristopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[143] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *International Conference on Machine Learning (ICML)*. Atlanta, USA, June 2013.

[144] Matteo Gadaleta and Michele Rossi. "IDNet: Smartphone-based gait recognition with convolutional neural networks". In: *Pattern Recognition* 74 (2018), pp. 25–37.

[145] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification". In: *Journal of machine learning research* 9.Aug (2008), pp. 1871–1874.

[146] Naomi S Altman. "An introduction to kernel and nearest-neighbor nonparametric regression". In: *The American Statistician* 46.3 (1992), pp. 175–185.

[147] Yichao Wu and Yufeng Liu. "Robust truncated hinge loss support vector machines". In: *Journal of the American Statistical Association* 102.479 (2007), pp. 974–983.

[148] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[149] Carl Edward Rasmussen. "Gaussian processes for machine learning". In: *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[150] Angelo Furno et al. "A tale of ten cities: Characterizing signatures of mobile traffic in urban areas". In: *IEEE Transactions on Mobile Computing* 16.10 (2017), pp. 2682–2696.

[151] Dan Hendrycks, Mantas Mazeika, and Thomas G Dietterich. "Deep anomaly detection with outlier exposure". In: *International Conference on Learning Representations (ICLR)*. Vancouver, BC, Canada, Apr. 2018.

[152] Sigurdur Sigurdsson et al. "Outlier estimation and detection application to skin lesion classification". In: *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*. IEEE. Orlando, FL, USA, May 2002.

[153] Yixin Chen et al. "Outlier Detection with the Kernelized Spatial Depth Function". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (2009), pp. 288–305.

[154] M. Lauridsen. "Studies on Mobile Terminal Energy Consumption for LTE and Future 5G". PhD thesis. Aalborg University, Jan. 2015.

[155] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. Tech. rep. 3GPP TS 36.213 version 10.1.0 Release 10, APR. 2010. URL: `http://www.3gpp.org.`.

[156] *TS 38.300, 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; NR and NG-RAN Overall Description*. Tech. rep. 3GPP, Jan. 2019.

[157] *UE Power Consideration based on Days-of-Use*. Tech. rep. Qualcomm Incorporated, R1-166368, Aug. 2016.

[158] S. Rostami et al. "Novel Wake-up Signaling for Enhanced Energy-Efficiency of 5G and beyond Mobile Devices". In: *Proc. IEEE Globecom 2018*. Dec. 2018, pp. 1–7.

[159] S. Rostami et al. "Optimized Wake-up Scheme with Bounded Delay for Energy-Efficient MTC". In: *Proc. IEEE Globecom 2019*. Dec. 2019, pp. 1–6.

[160] S. Rostami et al. "Wake-Up Radio based Access in 5G under Delay Constraints: modeling and Optimization". In: *IEEE Transactions on Communications* (2019; accepted, to appear).

[161] J. Wang et al. "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach". In: *Proc. IEEE INFOCOM 2017*. May 2017, pp. 1–9. DOI: `10.1109/INFOCOM.2017.8057090`.

[162] Amin Azari et al. *User Traffic Prediction for Proactive Resource Management: Learning-Powered Approaches*. 2019. arXiv: `1906.00951 [cs.NI]`.

[163] Y. Shu et al. "Wireless traffic modeling and prediction using seasonal ARIMA models". In: *Proc. IEEE ICC 2003*. Vol. 3. May 2003, 1675–1679 vol.3. DOI: `10.1109/ICC.2003.1203886`.