Machine Learning to Infer User Behavior in 5G Autonomic Networks

Illyyne Saffar

UNIVERSITÉ DE RENNES 1

NOKIA Bell Labs

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

Illyyne SAFFAR

Machine Learning to Infer User Behavior in 5G Autonomic Networks

Thèse présentée et soutenue à Rennes, le 10 Novembre 2020
Unité de recherche : Nokia Bell Labs & IRISA Rennes

Rapporteurs avant soutenance :

Ghaya REKAY-BEN OTHMAN	Professeur : Telecom Paris et Institut Polytechnique de Paris, France
Thi-Mai-Trang NGUYEN	Maître de conférences HDR : LIP6, Sorbonne Université, France

Composition du Jury :

Président :	Adlen KSENTINI	Professeur, Eurecom, Sophia-Antipolis
Examinateurs :	Nathalie MITTON	Directrice de Recherche, INRIA Lille-Nord Europe, France
	Thi-Mai-Trang NGUYEN	Maître de Conférences - HDR, LIP6, Sorbonne Université
	Ghaya REKAY-BEN OTHMAN	Professeur, Telecom Paris et Institut Polytechnique de Paris
	Zwi ALTMAN	Ingénieur de recherche, Orange Labs Chatillon
Dir. de thèse :	César VIHO	Professeur, IRISA-Université de Rennes 1
Co-dir. de thèse :	Marie Line ALBERI MOREL	Ingénieur de Recherche, Nokia Bell Labs
	Kamal Deep SINGH	Maître de Conférences, Laboratoire Hubert Curien Université Jean Monnet Saint- Etienne

Acknowledgement

I would like to extend my sincere thanks to all people who have contributed to the successfulness of this thesis. Firstly, I wish to express my sincere gratitude to my supervisors Marie Line Alberi Morel, Kamal Deep Singh and César Viho for their guidance, encouragement, insightful thoughts, exceptional patience and for the stimulating discussions as well as the sleepless nights we were working together before deadlines. It was my great pleasure to work with them along these three years. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisors and mentors for my PhD study.

Besides my advisors, I would like to thank all my colleagues in Nokia Bell Labs France for all the fun we have had in the last four years, for the board games, quizzes in breaks, etc. Thanks to them I have never faced Monday morning blues. My work was not just about PhD work and long hours of research but it was extra fun, yummy food and chocolate, and long gossip sessions too. Thanks for adding your magic touch to my otherwise boring and dull work-life. My special acknowledgment to CNTP-A team in ENSA Lab, to CNTP-A manager Alberto Conte and to CNTP manager Laurent Roullet for their understanding and for their time.

I would also like to thank, my open space neighbor, senior research engineer, Frederic Faucheux for his continuous help and for his generous support psychologically as well as technically with the hardware for a major part of the thesis and his patience for all the mistakes and the clumsiness that I have committed. Last but not the least, I would like to thank my family: my parents,

my dear husband, my brothers and sisters, my friends, and my cats for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.
Illyyne Saffar

Table of Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

1G First Generation of Cellular Network

2G Second Generation of Cellular Network

3G Third Generation of Wireless Mobile Telecommunications Technology

3GPP 3rd Generation Partnership Project

3SS 3 Step System

4G Fourth Generation of Broadband Cellular Network Technology

5G Fifth Generation Technology Standard for Cellular Networks

A-DA Artificial Data Augmentation

ADALINE ADAptative LINear Element

AI Artificial Intelligence

AODE Averaged One-Dependence Estimators

API Application Programming Interface

BBN Bayesian Belief Network

BGM Bayesian Gaussian Mixture

BN Bayesian Network

CART Classification and Regression Tree

CDF Cumulative Distribution Function

CDMA Code Division Multiple Access

CHAID CHi-squared Automatic Interaction Detection

CML Classical Machine Learning

CNN Convolutional Neural Network

CQI Channel Quality Indicator

CQI Channel Quality Indicator

CT Co-Training

CTL Cluster-Then-Label

CV Computer Vision

D2D Device to Device

DL Deep Learning

DNN Deep Neural Network

EI Expected Improvement

EM Expectation Maximisation

eMBB enhanced Mobile Broadband

eNB evolved Node B

EPD Extracted Portion Data

ETSI ZSM ETSI Zero-touch network and Service Management

FDA Flexible Discriminant Analysis

FNN Feedforward Neural Network

GBM Gradient Boosting Machines

GBRT Gradient Boosted Regression Trees

GNSS Global Navigation Satellite System

GPS Global Position System

GPU Graphical Processing Unit

GSM Global System for Mobile communications

ID3 Iterative Dichotomiser 3

IOD Indoor Outdoor Detection

IoT Internet of Things

KNN K-Nearest Neighbor

LARS Least-Angle Regression

LASSO Least Absolute Shrinkage and Selection Operator

LDA Linear Discriminant Analysis

LOESS LOcally Estimated Scatterplot Smoothing

LSTM Long Short Term Memory

LTE Long Term Evolution

LVQ Learning Vector Quantization

LWL Locally Weighted Learning

MARS Multivariate Adaptive Regression Splines

MDA Mixture Discriminant Analysis

MDS Multi Dimensional Scaling

MDT Minimization of Drive Tests

MI Mobility Indicator

ML Machine Learning

MRN Multilinear Relationship Network

MSP Mobility Speed Profiling

MTL Multi-Task Learning

NCID Network Clear Indication Delay

NCR Number of Cell Res-selection

NIPS Neural Information Processing Systems

NLP Natural Language Processing

NN Neural Network

OLSR Ordinary Least Squares Regression

OneR One Rule

PCA Principal Component Analysis

PCR Principal Component Regression

PLSR Partial Least Squares Regression

QDA Quadratic Discriminant Analysis

QoE Quality of Experience

QoS Quality of Service

ReLU Rectified Linear activation Unit

RIPPER Repeated Incremental Pruning to Produce Error Reduction

RNN Recurrent Neuronal Network

RS Reference Signal

RSRP Reference Signal Receive Power

RSRQ Reference Signal Receive Quality

SGD Stochastic Gradient Descent

SgT Single Task

SMOTE Synthetic Minority Oversampling TEchnique

SOA State of the Art

SOM Self Organizing Map

ST Self Training

SVM Support Vector Machine

TA Time Advance

TCR Time Period

TDMA Time Division Multiple Access

TL Transfer Learning

TPE Tree Parzen Estimator

UE User Equipment

UED User Environment Detection

UED User Mobility Detection

URLLC Ultra-Reliable Low LatenCy

VoD Video on Demand

VoIP Voice over Internet Protocol

WLAN Wireless Local Area Network

ZeroR Zero Rule

Notation

This section provides a concise reference describing notation used throughout this document. If you are unfamiliar with any of the corresponding mathematical concepts, [1] describes most of these ideas in chapters 2–4.

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix

Sets

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$[a, b[$	The real interval excluding b but including a

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 0
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\int f(\boldsymbol{x})d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x})d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Probability and Information Theory

$P(\mathrm{a})$	A probability distribution over variable
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x}\sim P}[f(x)]$ or $\mathbb{E}f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$D_{\mathrm{KL}}(P\|Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1 + \exp(-x)}$
$\zeta(x)$	Softplus, $\log(1 + \exp(x))$
$\|\|\boldsymbol{x}\|\|_p$	L^p norm of $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	L^2 norm of $\boldsymbol{x}$

Sometimes we use a function f whose argument is a scalar but apply it to a vector or matrix: $f(\boldsymbol{x})$ or $f(\boldsymbol{X})$. This denotes the application of f to the array element-wise. For example, if $\mathbf{C} = \sigma(\mathbf{X})$, then $C_{i,j,k} = \sigma(X_{i,j,k})$ for all valid values of i, j and k.

Chapter 1

Introduction: Context & motivation

During the past four decades, wireless mobile communication networks have seen four different generations. At the beginning of the 1980s, it began with the birth of the first generation (1G) of mobile networks. 1G systems used analog technology and only provided voice services. In the 1990s, the second generation (2G) of mobile networks emerged. 2G systems differed from the previous generation in using the digital mobile communication systems, and using the Time-Division Multiple Access (TDMA) scheme to support more users. 2G provided voice services and low rate data services (only 14.4 Kbps) [2]. As the use of 2G phones became more widespread and people began to use mobile phones in their daily lives, it was clear that the need for data services (such as access to the internet) became important. Hence, we witnessed the birth of the third-generation (3G) mobile networks. 3G systems were characterized by the Code Division Multiple Access (CDMA) technique. 3G was able to support mobile internet services with data rates up to 14 Mbps [2]. After 3G, we saw the arrival of 4G. Omnipresent, the 4th generation cellular networks are deployed all over the world and provide all-IP broadband mobile access and enriched video-based services to end users. With a rising tide of Internet of Things (IoT), more and more mobile smart devices are able to access the mobile Internet. Today, the whole world is witnessing the arrival of the fifth generation (5G) of

mobile networks and waiting for beyond 5G. 5G is revolutionizing the wireless mobile communication networks. The work described in this thesis is about research on 5G and beyond 5G.

This chapter is an introductory chapter where we detail the research context. We explain the motivations behind the work as well as the challenges to be overcome. We also list, per section, the problematic and the contributions made during this thesis. Afterwards, we present the structure of the document for the next chapters as well as the list of publications. This introductory chapter is organized as follows: **Sec. 1.1** provides a description of the global context and motivates our interest in cognitive autonomic networks for 5G. The problematic to deal with cognition and "inference" is presented by answering the question: how to extract cognition and intelligence from user data in order to anticipate the user needs or preferences so as to positively impact 5G networks and services? **Sec. 1.2** describes the different visions of the user behavior and its definition in the field of wireless networks, within the literature. The section also presents our proposed approach to define a model of user behavior. The proposed model allows to anticipate the needs and preferences of users. **Sec. 1.3** is about inferring user behaviour. It describes the problems that need to be solved, the challenges to overcome as well as our proposed solutions.

1.1 Context, motivation and main challenges

New 5G networks are promising many new services, new technologies and innovations such as enhanced mobile broadband (eMBB), ultra-reliable low latency service (URLLC), Internet of things (IoT), autonomous cars, Device to Device communication (D2D) , etc [3]. Actually, they are designed to ensure that “Every thing is connected to every thing at any time” by supporting both human-to-human and machine-to-machine communications, connecting up

to trillions of devices and reaching formidable levels of complexity and traffic volumes.

Such revolution brings a new set of challenges for managing the network otherwise due to its diversity and big size. So, the coming of 5G triggers a need for a radical change in terms of end-to-end management and orchestration of network and services. The direction is towards a full end-to-end automation for 5G and beyond [4],[5]. Hence, it will be necessary for 5G networks to largely manage themselves and deal with organisation, configuration, security, and optimisation issues. That is the goal of the ETSI Zero touch network and Service Management (ETSI ZSM) standard that addresses 5G as Autonomic Networks. The idea is to design a thinking part or a brain of mobile networks [6],[7]. It shall integrate computational systems that are able to manage themselves within an acceptable period of time with no or minimal human intervention. ETSI ZSM specifies the service as well as the network management and operations within flexible management framework that enables complete and autonomous operations without external intervention (called zero-touch management) [8], [9], [10].

So, the complexity, flexibility, dynamicity and the diverse prerequisites of 5G networks require that the network must be managed in an autonomic manner. To reach this goal, the automation of 5G targets the areas of end-to-end network slice management, monitoring, planning. The idea is to make network components capable of autonomous operation and serve their purpose without external intervention even in case of changes in the network and radio environment. The autonomous management includes the functions of self-configuration, self-repair and self-optimization. It brings benefits such as the simplified management of 5G network and a reduction in its deployment cost.

On the other side, with every advance in the generation from 3G to 4G and reaching 5G, users (customers) are more and more asking for better services,

with the highest possible quality and the lowest cost. It is expected from the next evolution of mobile networks that they will accommodate the ever-growing user-demands, services and applications and will guarantee a better Quality of Experience (QoE). One of the ways that can satisfy the users' expectation is by adding in-advance situational awareness to the network. This will make the network aware of the situations in which mobile users consume, or prefer to consume, their services and applications. By anticipating user preferences and needs, the networks will be able to efficiently face the variable consuming habits of users that impact the network conditions. This will allow the network to optimize every connection for every end-user, or device. Mainly, this will shift the objectives of network self-management and decision-making processes from a network-centric to a more user-centric view. Thus, the use of external additional knowledge on the mobile users' behavior that is defined in terms of habits of service usage will help networks to be more intelligent and more aware of the environment in which they operate. Indeed by being aware of user habits, 5G networks will efficiently face the variable consuming habits of users that impact the network conditions. *For this, the cognition based on user behavior knowledge can be considered as a key to provide intelligent and suitable networking solutions for 5G networks.* That way, they will succeed to satisfy end-users or consumers when faced with the increasing complexity of network management combined with numerous new applications and their heterogeneous requirements.

However, inferring the service consumption habits of users represents a very complex problem. This is because, inferring has to be done in a composite environment consisting of multi-services and billions of consumers or end-users. Automatic inference of mobile user behaviour is particularly challenging. Indeed, such inference should be done without requiring constant user interaction or without using personalized and refined questions. This is to avoid

exponential complexity. The first challenge is about system and methods that extract such cognition requiring minimal human intervention and minimizing system complexity.

Problematic 1

How to automatically extract and process the knowledge of the mobile user behavior, in order to anticipate the user preferences, and needs, with the goal to optimize 5G networks?

Contribution 1

The extraction of user-centred knowledge for anticipating the user needs and preferences to enhance 5G can be done within the global system depicted in the figure 1.1. The whole process is described in a 3-step system (3 Step System or 3SS) that is composed of the following 3 stages: 1) Data Collection, 2) Data processing for user behavior inference and user profile building and 3) Information usage for 5G optimization. To automatically extract the cognition about the user behavior, we propose to use the current promising techniques from the domain of Artificial intelligence (AI) and more precisely Machine Learning (ML) and Deep Learning techniques (DL). With AI, a machine mimics human minds and "learns" automatically from the environment, then it solves problems by maximizing the success probability.

Let's have a deeper focus on the 3 Step System:

1. The first step is dedicated to data collection. This step can be considered as the most delicate one among the 2 others since data is crucial for artificial intelligence solutions. That is to say, it can be seen as the

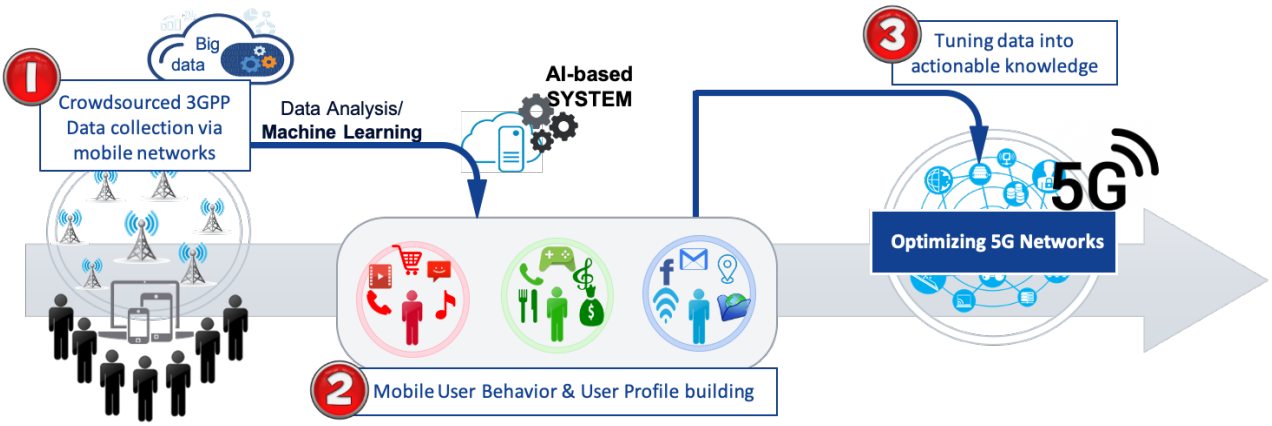

Figure 1.1 – 3 Step System for knowledge extraction in order to anticipate user needs and optimizing 5G. Step 1: Data Collection. Step 2: Data processing for user behavior estimation and profiling. Step 3: 5G optimization.

cornerstone guaranteeing the success of the 2 other steps. A detailed description of the data features and the collection campaign will be given in chapter 3, in section 3.2. The data collection step aims to collect data correlated to the user behavior or the manner in which the user consumes a requested mobile service. As a consequence, we have privileged a particular data collection mode referred as crowd-sourcing for building the training dataset. The crowd-sourcing mode consists in collecting signals measured directly inside the mobile phones. This mode allows to gather data which closely reflects the real life of a user. Such data captures all the diverse situations during the real activities of a given mobile user. Consequently, crowd-sourced data is highly representative of real conditions and, with machine learning techniques, we can expect a precise learning of the mobile user behavior in terms of their habits.

2. The collected data from the first step will serve as input to the second step. This second step aims to estimate the user behavior to build user profiles, in automatic way with minimal human intervention. For that, the data will be processed through Machine Learning and Deep Learn-

ing approaches. AI techniques target to design intelligent machines and software that can efficiently and automatically solve complex problems, without being explicitly programmed to do so.

3. In the final step, the cognition extracted as user profiles will be turned into actionable knowledge. This in turn will be injected to the 5G autonomic networks to make them smarter and increase their awareness. It will help 5G networks to face the variable consuming habits of users that impact the network conditions. At the same time, the demands or needs of customers will be individually met, thus focusing on user satisfaction.

In this PhD thesis, we focus on step 1 and 2. The step 3 is out of scope of this study. However to address step 1 and 2, first an appropriate model of mobile user behavior has to be proposed, which is discussed in the following sections.

1.2 Mobile user behavior modelling

In literature, there exist a plethora of definitions of mobile user behavior in different use cases. They differ from one work to another.

Authors in [11] perform an analysis on mobile user data to understand general aggregated user behavior based on several parameters. The parameters are (i) the mobile data usage (ii) user groups (either alone or sharing experience with others) (iii) geographical regions (USA, Europe) and (iv) different time frames by studying the impact of special events like holidays or sports events (soccer game in Europe and Superbowl game in US) on the potential differences engendered on general trends of the daily user behavior. They analyzed aggregated user data traffic collected from gateways of different operators in different parts of the world. The investigation shows similarities and differences in different geographic regions, in different time frames and for different operators.

The comparative analysis is done on gateways in terms of average data rate (kbps) per session (user), total number of sessions, and total throughput in a specific context (Daily Usage (Weekdays and weekend), Special Days, Holidays). In works studying social networks, the correlation between the mobility patterns and the interests in requested content have been showed for users with social ties. In [12], authors consider the context (radio, resource and QoS metrics) as the relevant information defining the user behaviour for the interaction between the user and the serving Point of Attachment. They propose an intelligent handover mechanism which takes into account multiple requirements and various context information in order to reduce handover delays and guarantee session continuity. [13] uses the context awareness information to extract the network situation and reduce the traffic in between fronthaul and backhaul when subscriber authentication and authorization takes place. Precisely, authors aim to distinguish the normal from emergency situation using the user behavior information in order to deliver proactive control facing any situation. Their context information or user behavior definition is based on 2 features (i) user motion pattern and (ii) backhaul outage probability. They are extracted or inferred from from real-time information of devices (mobility and position) and some network information such as (coverage area, traffic model and backhaul throughput).

The above works demonstrate the positive impact of using cognition of the user behavior for enhancing network operations. However, in these investigations, the user behavior is mainly characterized by network metrics such as traffic data, QoS, resource, outage probability and mobility pattern. They do not give a direct vision of phone usage habits of users, but rather a vision of network conditions. There are other studies in literature which investigate the use context (where, what and when) to characterize the phone usage habits of users. As discussed in the following text, they evaluate how such characteri-

zation can positively impact network operations.

In [14], the authors demonstrate that the content-based caching network is remarkably more efficient when using the correlations between mobility and interests than random caching. [15] focuses on profiling the usage patterns of mobile applications and investigates how, where, and when smartphone applications are used from spatial, temporal, and user perspectives. Research in [16] investigated the current dynamics of the Malaysian market for smartphone and the usage behaviors of consumers. Furthermore, consumers' usage behaviors such as using a smartphone for email, web browsing, gaming, and document reading were examined. In [17] the authors describe the user situation with information of the place derived from location data, time, and other people surrounding the user. In [18], the context of use is seen as a joint impact of 4 items (i) smart mobile devices, (ii) mobile applications (iii) used information and communications services and finally (iv) the user's environment. With such description, the potential of context information can be seen in developing new, more personalized mobile services and applications. Authors in [19] define the context of use as the combination of the time and the place information (the user environment) in an attempt to put a meaning to a particular place/environment while consuming a service. In [20] authors show that knowing the user environment change in the few coming seconds can enhance a user's quality of experience for a video application using optimization of video data transfer. They accelerate the filling of application layer buffer to prevent coverage hole. Other researchers [21] show some improvements in user localization accuracy by using the knowledge on whether a user is indoor or outdoor. Works in [22] and in [23] show that user mobility states or pattern can help in the mobility optimization process. Such information is important for mobility management. Other works have investigated the impact of knowing the application type per user to enhance either the performance of Enhanced

Inter Cell Interference Coordination in interfered environment in [24] or either QoE in[25].

The above works represent a non exhaustive list of what network operations can benefit from the knowledge of user behavior. They illustrate the achievable positive gain. But we observe that only a part of the user situation is considered by the previous works to build the user behavior model. They mainly consider and study the impact of one or two features defining the user behavior separately on different specific network operations. This leads to specific definitions. To be able to better understand the habits of consumers or end-users, we need a complete definition of the user behavior. Such definition should encompass all the situations experienced by the users linked with the requested services or applications.

The [ISO 13407:1999] standard [26] highlights the requirements to know not only the context of use, but also information on users (demographics), system and applications because the context affects the service/application usage situation as well as the user and the used technology. The standard defines the context of use as the whole situation relevant to an application and its set of users. Therefore, it is given by the characteristics of the users, tasks and the environment in which the system is used. In [27] the authors define the context as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Consequently, the use context defines then the diverse situations where mobile users experience their services/applications while connected to mobile networks.

Definition 1: How we define the user behavior?

Extending the definition of context from [27], we therefore state that the user behavior model is the direct reflection of all the usage situations of mobile services, and the users themselves, which provides a picture of the consuming habits or preferences of users. The model considered shall be linked to the use context experienced by mobile users and depict all mobile user situations during which they consume mobile services or use varied applications.

The potential value of this definition that links the use context to the user behavior lies in its possibilities to predict noticeable trends in user behaviors/habits related to the different situations experienced by the mobile users, the given application and the users. However with this definition, inferring the use context of mobile users become a prerequisite to infer mobile user behavior. Today, assessing the use context remains a difficult task yet. It is particularly problematic when the objective is to automatically detect this context, targeting the personal usages or preferences prediction, without requiring constant user interactions through personalized and refined questions. Consequently, a reliable detection of mobile use context attributes, considering multiple user-situations without human intervention, is still an open issue and it is fully addressed in this PhD thesis.

We first propose a model of mobile user behavior that can abstract a large scope of situations in which a mobile user consumes or uses varied mobile services or applications. Then, we will address the challenge of reliable detection of mobile use context attributes in a non-intrusive way by considering Machine Learning techniques.

Problematic 2

How to properly define an appropriate model of the user behavior?

Contribution 2

We propose to model the user behavior as a QoE-influencing features model. The model is composed of the factors defining the contextual information that have an impact on QoE and that link the usage situation, the user and the application

To define an appropriate model, we take into consideration two major requirements.

— First, the user behavior model will be *an abstraction of the diverse usage situations experienced by mobile users* inside the delivery zones of mobile services. It will also be an abstraction of the users themselves as well as the application.

— Second, we target *to bring a positive impact on Quality-of-Experience (QoE)* with the intelligence extracted from user behavior model linking the usage situation and the application, while a user is using his phone. QoE is a metric that measures the mobile user's satisfaction depending on his or her experience of the service.

We propose also to model the user behavior with the factors defining the contextual information that have an impact on QoE and that link the usage situation, the user and the application, while a user is using his phone. The main contextual information is summarized in the table 1.1. It is made of four main categories of QoE-influencing factors: use context, application, system and demographics. All have direct link with QoE. The use context includes the environment, with more refined information like indoor/outdoor location (instead of only the exact user coordinates), device speed and orientation, the

social context, i.e., people around the user and the type of access medium. The applications used on mobile devices are diverse. They include communication, web, social network applications, multimedia streaming, TV channels applications and also mobile apps [28]. The demographics give the human characteristics as the age or the education level. The system characterizing the codec and the device features is out of scope of this work.

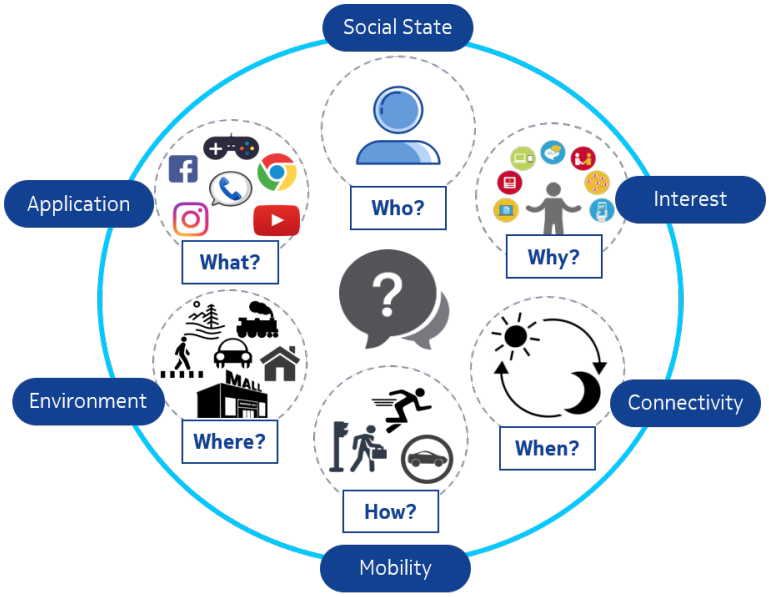

Figure 1.2 – User Behavior Modelling: Intelligence for anticipating User Preferences & Needs

In order to define the model taking into account the requirements, we will address 6 main questions (5W1H) based on Kipling's method [29]. The Five W's and One H Method represents a list of the six universal questions: What? Why? Who? Where? When? and How?. The essence of the method consists in a consistent formulation of these questions. Detailed, specific and original answers to these questions provide a more complete analysis of the problem. They open up additional opportunities and allow to formulate better solutions and decisions. Therefore, the user behavior is then modeled as a multi-attribute entity where each attribute is directly linked to the service usage and to the answer

Use context	Environment	Indoor (home, office, café) / Outdoor (incar, suburban, urban)
	Mobility/ Speed	Walking, driving, standing, sitting, ect.
	Access medium, radio interface	Wired, wireless, 3G/4G/5G, WLAN.
	Social	Alone, with a person, with a group, ect.
	Time/ Periodicity	Day time, holidays, special events, ect.
Application	Type	VoIP, live or non live service, Web access, VoD with short/full movies, on-line gaming, ect.
	Content/Interest	Action movie, interview, video-conference or video call, image, synthesis, ect.
System	codec features	encoding features, quantification parameter, resolution sampling rate, frame rate screen resolution, color depth, user interface capabilities, screen illumination and size,
	device	computational power, memory, battery life-time, iOS, hardware
Demo-graphics	user	occupation, education level, age, gender, social/task

Table 1.1 – Quality-Of-Experience-Influencing Factors of contextual information

of one of 5W1H questions. The model is depicted in Figure 1.2.

The 5W1H questions are defined as follows:

1. **Who** is the user?
   $\xrightarrow{\text{answer}}$ Demographics or social state (alone/ with a friend/ in a group/ etc.)
2. **Where** is the user while using his phone?
   $\xrightarrow{\text{answer}}$ Environment (indoor/ outdoor/ etc.)
3. **What** is he consuming? What type of traffic/application?
   $\xrightarrow{\text{answer}}$ Application (conversational/ video/ gaming/ etc.)
4. **When** is he using the phone? Mostly at night or at morning

$\xrightarrow{\text{answer}}$ Time and periodicity (Day time/ night/ frequently/ holidays/ special events/ etc.)

5. **How** is he using it? static or moving?
   $\xrightarrow{\text{answer}}$ Mobility State and Speed (low/ high/ medium/ speed range/ etc.)

6. **Why** to explain the relationship between all the other questions.
   $\xrightarrow{\text{answer}}$ Interest (news/ sport/ action movies/ etc.)

Hence the model is made of 6 attributes delivering information on preferences or habits of mobile user when he is connected to mobile network.

Thus, in this PhD thesis a user behavior model is established based on the rules defined by Kipling [29]. The model can be seen as an abstraction of contextual information of a mobile user defined by answering 6 main questions (Who, Where, What, When, How, Why). Each answer of a question corresponds to an attribute related to one of QoE-influencing features of mobile contextual information. In fact, enriching the autonomic networks with knowledge of use context of mobile service users, which in turn influences QoE, enables the network to make decisions that positively enhance the experience of the connected user. This is illustrated in [20], [22], [16], [24] and [25].

1.3 Data and Machine Learning driven detection of user behavior

Detection of context is not a recent challenge [7]. It has been studied since a decade from now. However, all studies that we presented previously in section 1.2 have investigated or evaluated the detection or the impact of only one or two attributes for one or more purposes. We can clearly imagine that if a global system is able to detect all the attributes of the user behavior model, at

same time, then it can provide a more enriched information to the network. As a consequence, the autonomic networks become more performing with more awareness of all user habits and preferences. They will be able to cope better with the changes and evolution of the way users consume mobile services. However, detecting the user behavior requires to answer all the questions at the same time since all the attributes impact the user behavior and are linked together. The combination of the 6 answers gives a complete view of the user behavior. Thus, the challenge is to find an unified system that is able to simultaneously deliver the answers to 6 questions, in a synchronous way:

— the social state of the user,
— his environment,
— his speed range or mobility,
— his traffic or application consumption,
— his time of consumption,
— and also his interest in consuming such a content in such environment, during that time, with that speed range.

However, finding the answers is complex and it is difficult to model them mathematically. This corresponds to a multi-objective optimization problem. Furthermore, it shall be done in a non-intrusive way and with a minimal cost. To find an appropriate solution to this problem the following question has to be answered.

Problematic 3

How to efficiently infer the mobile user behavior reflecting the real life of user, with minimal human intervention, low-complexity system and minimal response time?

To solve the problem, we fix some constraints and select the appropriated approaches.

Contribution 3

We propose to

1. opt for a Multi-Task neural networks architecture for ensuring a parallel inference of all attributes, in a unified system, considering a single model,
2. mutualize and minimize the number of data inputs required for simultaneous inference,
3. investigate Machine Learning approaches and more precisely Deep Learning approaches to automatically solve complex problems,
4. collect data using a crowd-sourcing mode to obtain training dataset having representative data about user behavior.

To sum up, figure. 1.3 shows our proposed system to achieve our global target for automatically inferring the mobile user behavior, based on the definition we proposed in section 1.2, under our imposed constraints. We adopt a Multi-Task neural network architecture to achieve the parallel inference and investigate the problem solving within a unified system. Such unified system ensures less complexity than using 6 different and separate systems (one system per attribute or per question to answer). So, it would reduce the processing / computational complexity, while avoiding redundant computations distributed over the separated and non optimized systems. In addition with the mutualization of the data inputs, the number of inputs or attributes per task can be reduced at the phase of signal selection. We will preferably select signals or data that can contribute to deliver the answers of more than one question. Consequently it would limit the complexity of the whole system.

The investigated solution for answering the questions (how and where?) is based on Machine Learning approaches and more precisely Deep Learning ap-

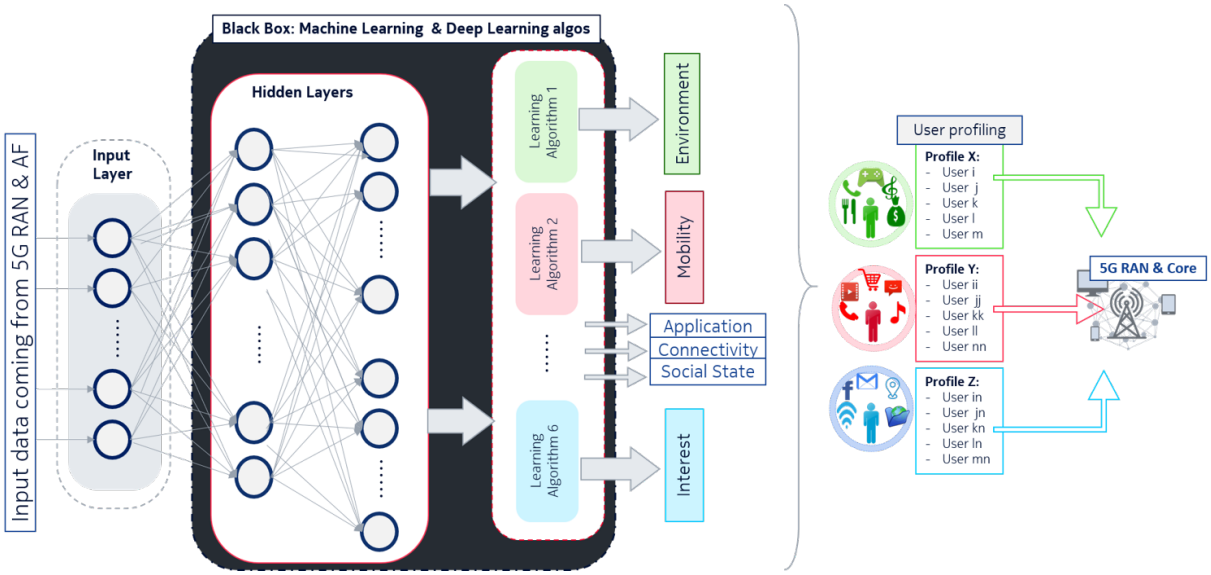

Figure 1.3 – Challenge: One AI-based System to infer the user behavior (Behaviors to infer: Environment, Mobility, Application, Connectivity, Social State, Interest)

proaches. They are indeed good candidates due to their ability to deal with complex problems and to characterize the inherent relationships between the inputs and outputs of a system without human involvement. However currently, Deep Learning (a sub field of Machine Learning) is gaining much popularity due to its supremacy in terms of accuracy when trained with huge amounts of data. As per Andrew Ng [30], the chief scientist of China's major search engine Baidu and one of the leaders of the Google Brain Project, "The analogy to Deep Learning is that the rocket engine is the Deep Learning models and the fuel is the huge amounts of data we can feed to these algorithms". Deep Learning's performance continues to improve when more and more data is used for training. This results in it outperforming the traditional models/algorithms of Machine Learning [31], [32]. Furthermore, increasing complexity of a given task, combined with complicated relationships between data, makes Deep Learning a good candidate. In this work, two learning categories, supervised or semi-

supervised, have been investigated and evaluated. In contrast to supervised learning which needs to know the output/labels as well as the input to build the model, the semi-supervised method mixes a supervised learning method with an unsupervised learning method. The latter in turn does not require the labels or the output. Such method is interesting because it diminishes even more the need of a human intervention because it requires less data to be labeled. We focus on inferring the mobile user behavior with Deep Learning algorithms in these two cases. The performance is also compared to those obtained using classical Machine Learning algorithms.

However, opting for Deep Learning adds another challenge to tackle in terms of *'Data'*. Actually, data is the fuel of any artificial intelligence approach. Without relevant data, we can't even think about Deep Learning. The better the data is, the better the solution will be. We choose to feed Machine learning tools with data collected in a crowd-sourcing mode [33]. It is defined as an online approach for problem solving through involvement of crowd. For crowd-sourcing approach, we consider that current smartphones are equipped with enough sensors to get fine grained context information. A set of significant number of smartphone users includes very large number of users with different user's characteristics. Considering the fact that users carry their smartphone devices with them daily, smartphone devices allow collecting the data about their owners. Considering the potential to monitor users of the smartphone devices, number of smartphone sensors and amount of user's data which can be collected, we find that smartphone devices present an excellent platform for collecting data about users.

Therefore, Machine Learning algorithms trained on datasets collected in crowd-sourcing mode allow to learn very diverse real-world situations.

1.4 Real life data: preferred situations, imbalanced and noisy data

Actually, contrary to traditional TV users who just watch scheduled programs, mobile service users are free to choose the content they want, at any point in time and space. During a day, a user can be in different situations, such as walking outdoor, in a car, at the work office, in a mall, in a café or at home. As a matter of fact, mobile users' preferences for certain applications or contents is linked with the usage situations [34]. Statistical studies show that mobile phones are mostly used in a building for internet service (80%) and for a call (70%) [35]. This can be explained by the fact that the different use contexts pose their own limitations, which in turn impact the potential application usages. In [28], [36], [37], the most commonly mentioned physical environments of application usage are again indoors (waiting halls or lounges, work, home and cafes), but also include vehicles, such as public transportation and private cars. In motion, audio is the preferred media, whereas, during stationary reception, text and video are the most pleasant media. For information assimilation, the café environment is preferred, while the bus or car environment is not that preferred. This is explained by the calm and the pleasant atmosphere of a café, which is suitable for focusing on viewing. In the bus context, people may focus their attention to, e.g., watching mobile TV, but a complicated task while moving generally results in an unpleasant entertainment experience. The mobile devices are mostly used in indoor or in-car context.

With respect to users' routines, the requested applications are used at specific moments during the day, either in morning, or in evening before going to sleep, in the vehicle, and inside/outside the office. They can also be used all day depending on age or social status of users. Typically, people do not watch mobile TV during short journeys and in noisy environments. In such situations

they prefer textual information over video. However, some types of users view mobile TV in different times of day, from morning to late in the evening. With respect to the social context, the applications were used 80% of the time, on average, when the user was alone. Motivations for individual viewing are killing time, fighting loneliness, keeping up-to-date, browsing content and unavailability of other possibilities to watch television. In public locations, mobile TV also serves to build one's own space and privacy. Even though screen size and use of headphones lead to practical limits for watching same content among multiple viewers, sharing the same device, mobile TV is also used in shared situations to entertain children, to broadcast a piece of information. Thus, such application enables both an individual and a shared viewing experience.

As a matter of fact, mobile users' preferences for certain applications or contents is linked with the usage situations [28],[34]. People spend most of their time indoor than in mobility and outdoor. Consequently, outdoor data is less represented than indoor data when considering phone usage.

Furthermore, the dataset built in crowd-sourcing mode is inherently not strictly controlled during the data collection. In our case, it suffered mainly from a problem which is related to the signal recording tool. This anyway serves an example of the problems that one may face. To automatically record various cellular signals or data during a given time slot, we employed a mobile application. The recorded values are stored in files on the mobile or directly sent to a data platform. But, during short disconnection events, this often (1) leaves an empty value in the signal vector to follow or (2) fills it with random or default values or (3) duplicates the same value several times; this depends on the signal in question.

So, we also expect that the system has to process imbalanced and noisy data. Knowing that imbalanced and noisy data presents a big issue for ML models, different solutions will be investigated to cope with this constraint.

Problematic 4

How to deal efficiently with real data which is imbalanced and noisy?

Contribution 4

We propose to automatically detect duplicates, anomaly and missing part in the data or signals recorded. A post-processing of data cleaning after recording has to be performed to remove or replace false values and imbalanced data, etc.

Noisy data implies that we need to do data cleaning. It is important to obtain good performance for the models build using learning to do environment or mobility state detection. To handle this issue, an analyse of general properties of the input data (called also features) allows us to efficiently detect missing data and outliers which, however few they are, considerably degrade the performance of the task asked from the learned models. These have been replaced by more significant values.

Imbalanced classes is a common issue for ML classification approaches where the classes are not equally represented. For example, for the anomalies detection use case, we face this issue since the anomalies are scarce as compared to normal events. In our case, the real data collected using crowdsourcing mode requires also to tackle this problem. Sometimes, this imbalance is also a by-product of data cleaning, which has to be done to overcome the noise inherent to real data. Actually, different techniques can be found in literature to solve it [38]. A simple and a common approach, which we adopted in this thesis, is to artificially over-sample the minority classes (or artificially under-sample the abundant classes) in the dataset.

1.5 Summary of Thesis Contributions

In this thesis, we will present an investigation of user behavior modeling. The user behavior model considers two QoE-influencing features related to mobile use context: the user environment and mobility state. The environment and mobility are important factors regarding QoE since they have a big influence on QoE. For example, a user who is indoor would experience a very different service quality as compared to the one who is outdoor, all else being equal. Actually, the environment and the mobility together set up the conditions in which a mobile user consumes the requested services and applications. Different from the state of art, we answer simultaneously to the following questions, using Deep-Learning-based models: how and where a mobile user consumes the mobile services? The detection of user behavior using the proposed model is done with minimal human intervention. We also consider multiple and varied real situations where users are being connected to cellular networks, while experiencing a service or using an application. To provide a high-performance learning model, the model design is done sequentially by first separately considering the detection of each attribute and then by conjointly inferring them at the same time. Furthermore the training is done using the data which is collected directly by smartphones using crowd-sourcing approach. The performances obtained with Deep Learning algorithms based on either supervised, semi-supervised (semi-hybrid systems) or Multi-Task approaches have been studied and evaluated in case of simultaneous inference of both the environment and the mobility. They have been also compared to classical Machine Learning algorithms. All Deep Learning models have been trained and tested using real radio data that is signal measurements collected within a 4G network. They represent 3GPP defined signals or indicators measured by UE and sent to eNB via standardized protocols.

In the following section, we define the structure of the thesis and subsequently highlight the contributions of each chapter.

1.6 Document structure

The thesis report is made of seven chapters including this one. Additionally, there is also a list of figures, a list of tables, a list of abbreviations, mathematical notations, bibliography and appendices. In the following, we present a brief description of the content of each chapter.

- **Chapter 2: Machine Learning: Definitions & Generalities**
  Describes, the basic concepts of Machine Learning and Deep Learning. Beginning from the learning types then passing by the machine learning algorithm families that we have categorized into two categories: Classical Machine Learning algorithms (or shallow algorithms) and deep learning algorithms.
- **Chapter 3: Machine Learning Workflow Toward User Behavior characterisation**
  hapter discusses the workflow of any Machine Learning based-solution that includes all the required steps to build the proper machine learning solution from scratch. We propose a 5-step workflow to fix the basic points of the process adopted to solve the user behavior modeling: 1) Identifying the problem and analyzing the needs 2) Collecting and gathering varied data 3) Cleaning and preparing the data 4) Model choice and training 5) Scaling-up, optimization and evaluation. Then an explanation on how the data used for our purpose has been generated.
- **Chapter 4: User Environment Detection: Where is the mobile user while experiencing a service?**

It investigates the environment detection of an active mobile phone user, using supervised and hybrid/semi-supervised Deep Learning based methods. We will empirically evaluate the effectiveness of the investigated methods using new real-time radio signals and with complete, as well as partial, ground truth information. A comparison will be done with classical ML methods. Data is gathered massively from multiple, typical and diversified locations (indoor and outdoor) of mobile users. In a further investigation, we propose to improve the classification granularity by detecting more than two classes (Indoor/ Outdoor), which is up to five classes corresponding to five relevant environment categories. Relevant multi-class schemes are proposed to efficiently regroup the multiple environment categories in more than two classes.

- **Chapter 5: Mobility Speed Profiles (MSP) Detection: How is the user when experiencing a service? Static or Moving?**

  It investigates the mobility speed range detection of an active mobile phone user using supervised Deep Learning based methods. We empirically evaluate the effectiveness of our approach using real-time and highly representative radio data that best captures the real daily movements of users. This data includes ground truth information and the whole dataset has been gathered massively from many diversified mobility situations. In a further investigation, we will propose to go further in granularity by detecting more than three classes (Low, Medium or High), by going up to eight classes corresponding to multiple relevant situations. Relevant multi-class schemes are proposed to efficiently regroup the multiple mobility speed range categories.
- **Chapter 6: Multi-Task Learning for joint detection of environment and mobility**

Deals with the joint detection of the first two user behaviour attributes which impact QoE. They will be already investigated in previous chapters. In order to simultaneously answer both questions: where is the user while consuming a service and how is he consuming it, we propose a Multi-Task based Deep Learning architecture. For such architecture we use mutual data or inputs to infer the two attributes at the same time. For performance evaluation, we have used the same dataset used in investigating both of the mobility speed profiling and the environment detection. Such real-time radio data has been massively gathered from multiple diversified situations of mobile users.

- **Chapter 7: Conclusion**
  This chapter draws conclusions and proposes directions for further research

1.7 Publications list

This thesis led to 5 publications that are listed in the following:

1.7.1 International Conferences with review committee

The conferences' publications list in a chronological order is as follow:

1. **Machine Learning with partially labeled Data for Indoor Outdoor Detection [39]**
   — Conference: IEEE Consumer Communications & Networking Conference (CCNC)
   — Location: Las Vegas - USA
   — Date: Jan. 2019
2. **Semi-supervised Deep Learning-based Methods for Indoor Outdoor Detection [40]**

— Conference: IEEE International Conference on Communications (ICC)

— Location: Shanghai - China

— Date: May. 2019

3. **Mobile User Environment Detection using Deep Learning based Multi-Output Classification [41]**

— Conference: IFIP Wireless and Mobile Networking Conference (IFIP-WMNC)

— Location: Paris - France

— Date: Sep. 2019

4. **Deep Learning based Speed Profiling for Mobile Users in 5G Cellular Networks [42]**

— Conference: IEEE Global Communications Conference (GLOBECOM)

— Location: Hawaii - USA

— Date: Dec. 2019

1.7.2 International Workshops with review committee

The workshop papers list in a chronological order is as follow:

1. **Multi-task Deep Learning based Environment and Mobility Detection for User Behavior Modeling [43]**

— Conference: The International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)

— Location: Avignon - France

— Date: Jun. 2019

Chapter 2

Machine Learning: Definitions & Generalities

Social Networks, Videos, Films, Photos, Smartphones, Connected Objects, etc, make us and our world overwhelmed with data. The amount of data in the world, in our lives, seems to go on and on increasing and there is no end in sight. Omnipresent, Mobile Networks also are growing faster every day, reaching formidable levels of performance but also of traffic and data volume and making the task of managing them as well as analyzing them prodigious. As matter of fact, the better these data are analyzed, processed and then used, the smarter and the more powerful the networks will become, and this is essentially due to the ascendance of the artificial intelligence fields nowadays. Data processing, Information exploiting, Classification, Regression, Clustering, Machine Learning, Deep Learning, Artificial intelligence they are all a buzz words that we hear in every day life. These terms are often used too broadly, synonymously, or simply incorrectly. Actually, the border distinguishing every field is so blurry as we can see in figure 2.1. Data processing, Information exploiting, Classification, Regression, Clustering, Machine Learning, Deep Learning can all be considered as sub-fields of artificial intelligence in computer science that give computers the needed autonomy to learn or to make decisions without being explicitly programmed. While practically useful, these techniques are not very widely and properly deployed in the wireless networking field compared in

speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Rare are papers that mention Mobile Networking field as a direct application domain of these techniques [44], [45]. Meanwhile, a number of surveys on machine learning and neural network applications in wireless networking have emerged lately with 5G researches. However, they are kind of limited in terms of application fields or in presenting a concrete guidelines on how, when, and where to use different machine learning tools in the context of wireless networks [46].

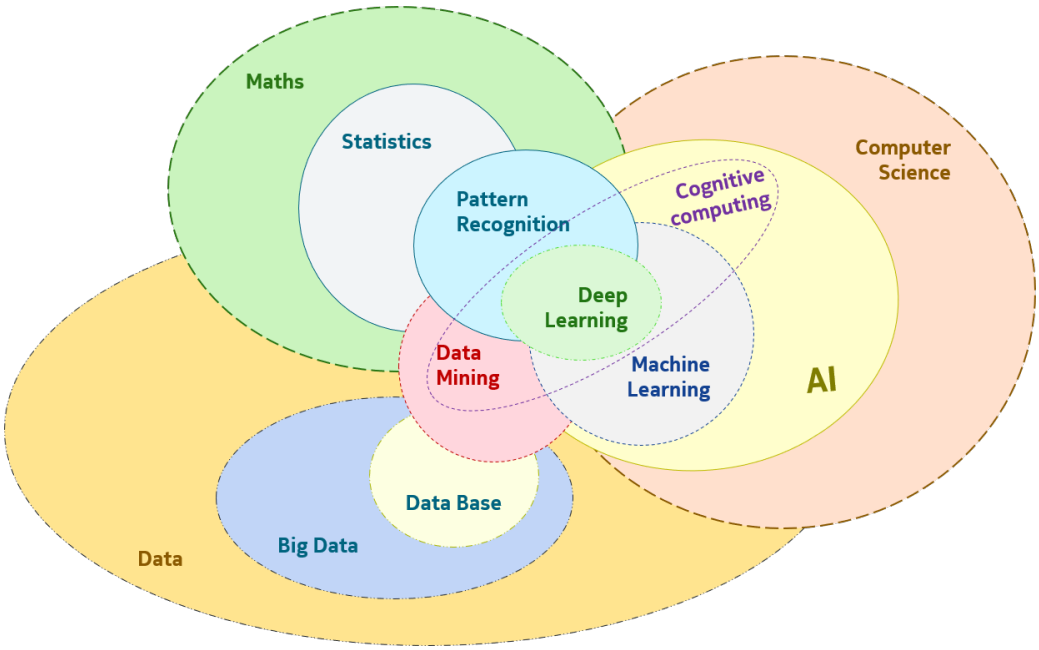

Figure 2.1 – Machine Learning Fields

This chapter aims to give, at first, an introduction to machine learning and Deep Learning in general and then, a description of the methodology adopted to develop the ML model for inferring the user behavior. It is organized as follows. **Sec. 2.1** provides an overview of Machine Learning with their definitions, the learning categories and the main families of ML algorithms. In **Sec. 2.2** we detail the existing machine learning algorithm families that we have categorized into two categories: shallow or classic algorithms and deep learning algorithms.

Section **Sec. 2.3** presents the conclusion of this chapter

2.1 Learning types

Machine Learning (ML) is the concept of bringing the digital world closer to the real world by training systems, computers, machines, in order to become more intelligent, to make decisions, to estimate the unknown and to predict the next steps in any ambiguous context. Technically, ML categorize large data sets by observing their regularities and recognizing their patterns. This can be done through a number of Machine Learning algorithms which learns from past and present events and then adapt the environment, regardless of human interventions.

One question that always arises every time we talk about ML is "When to Use Machine Learning?". Indeed, there is no “one size fits all” answer to this question but Machine Learning can be seen as an efficient alternative to the conventional engineering flow when problems appear to be too complex to be studied in its full generality or to be modelling mathematically or once we face many cases to satisfy. Authors in [47] identify a non exhaustive list of criteria that help distinguish when ML approaches are recommended to be used, for example we cite:

1. The task involves a function that maps well-defined inputs to well defined outputs: eg. Classification of spam and non Spam mails
2. Large data sets exist or can be created containing input-output pairs
3. The task provides clear feedback with clearly definable goals and metrics
4. The task does not involve long chains of logic or reasoning that depend on diverse background knowledge or common sense
5. The task does not require detailed explanations for how the decision was made

6. The task has a tolerance for error and no need for provably correct or optimal solutions
7. The phenomenon or function being learned should not change rapidly over time
8. No specialized dexterity, physical skills, or mobility is required

Thus, it is important to remember that ML is not a solution for every type of problem. There are certain cases where robust solutions can be developed without using ML techniques. For example, you don't need ML if a target value can be determined by using simple rules, computations, or predetermined steps that can be programmed without needing any data-driven learning [48]. ML is mainly used when the studied task is too complex to be modelled mathematically, it is also used to automate tasks. ML is a good candidate to analyse or predict the user behavior which is a complex task that can not be easily modelled mathematically. Globally, using ML to infer the user behavior can be seen as the first step of automating the decision making process of Autonomic Network.

We can distinguish three different main types of Machine Learning problems: supervised learning, unsupervised learning and Reinforcement Learning which are briefly introduced below. In this thesis we are rather interested in supervised and semisupervised learning which is the combination of supervised and unsupervised learning (mix of tagged and untagged data).

2.1.1 Supervised Learning

When the machine learning is trained using for every input a corresponding output or label or target or also called tag or class, so it is therefore a supervised learning. Thus, the supervised Learning is based on on apriori knowledge. The objective is, using some training data, to deduce a function that makes the

best possible mapping between inputs and output (see figure 2.2). The training data consists of tuples $\{ (\boldsymbol{x}_i, \boldsymbol{y}_i) \mid (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{X} \times \mathbb{Y} , i \in [1, N]\}$, where

— $\boldsymbol{X} \in \mathbb{X}$ is a set of observations composed of a random vectors $\boldsymbol{x}_i$ of dimension p ($\boldsymbol{x}_i = (x_{i_0}, x_{i_1}, ..., x_{i_p})$) that predict a certain output $\boldsymbol{y}_i \in \mathbb{Y}$

— N is the number of observations (samples) taken from $\mathbb{X}$

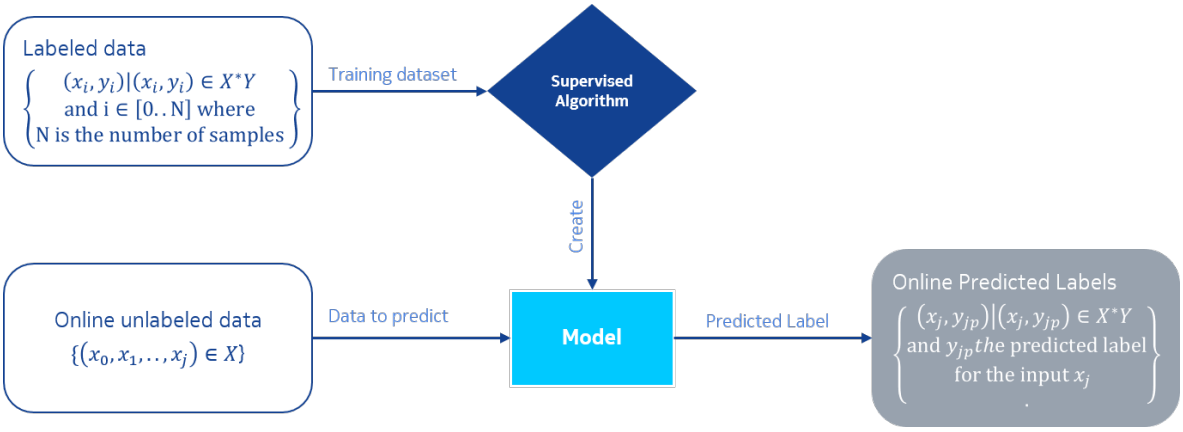

Figure 2.2 – Scheme of a supervised learning algorithm

The variable to predict $\boldsymbol{y} \in \mathbb{Y}$ can be a quantitative variable (as in the case of regression problems) or a qualitative variable (as in the case of classification problems). The objective of supervised learning is to find the unknown f function given some tuples $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}, f(\boldsymbol{x}))$. The The function f is an element of some space of possible functions $\mathbb{F}$, usually called the hypothesis space. An estimate $\hat{f}$ of the function f is obtained as the function that minimizes the empirical risk on the training set. The function of empirical risk has the following formula:

$$R_{emp}(\boldsymbol{w}) = \frac{1}{N} \sum L(\boldsymbol{y}_i, \hat{f}(\boldsymbol{x}_i, \boldsymbol{w}))$$

with

— L a loss function measure how well a function fits the training data. For training example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, the loss of predicting the value $\hat{\boldsymbol{y}}_i$ is $L(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)$

which measures how different the prediction $\hat{\boldsymbol{y}}_i$ from the true label or output $\boldsymbol{y}_i$.

— $\boldsymbol{w}$ a set of parameters that estimate $\boldsymbol{y}_i \in \mathbb{Y}$ as $\hat{\boldsymbol{y}}_i = \hat{f}(\boldsymbol{x}_i, \boldsymbol{w}) = \boldsymbol{x}_i \boldsymbol{w}$.

Once the trained model is build, it will be able to provide target/ tag/ output/ class for any new input after sufficient training. So during the learning phase the learning algorithm seeks a function from inputs to the respective outputs. There are two main types of supervised learning problems: they are classification that involves predicting a class label and regression that involves predicting a numerical value:

1. Classification: When inputs are divided into two or more classes and the learning algorithm must build a model that assigns new inputs to one of these classes (or two it depends of the study case).
2. Regression: Estimating relationships between variables.

2.1.2 Unsupervised Learning

Unlike in supervised learning, in this case there is no apriori knowledge. Here we do not have tuples $\{ (\boldsymbol{x}_i, \boldsymbol{y}_i) \mid (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{X} \times \mathbb{Y} \ , i \in [1, N] \}$, we simply have (without the target or the output $\boldsymbol{y}_i$):

$$\{ (\boldsymbol{x}_0, .., \boldsymbol{x}_i, \boldsymbol{x}_N) \mid (x_i) \in \mathbb{X} \ , i \in [1, N] \}$$

In an unsupervised learning problem, the objective is to model the structures or the underlying distributions of the data without referring to known or labeled results (see figure 2.3). It is called unsupervised because, unlike the supervised one, it tends to be more subjective since it does not have correct answers. Algorithms serve to discover and present interesting structures in the data.

There are many types of unsupervised learning, the most known ones are:

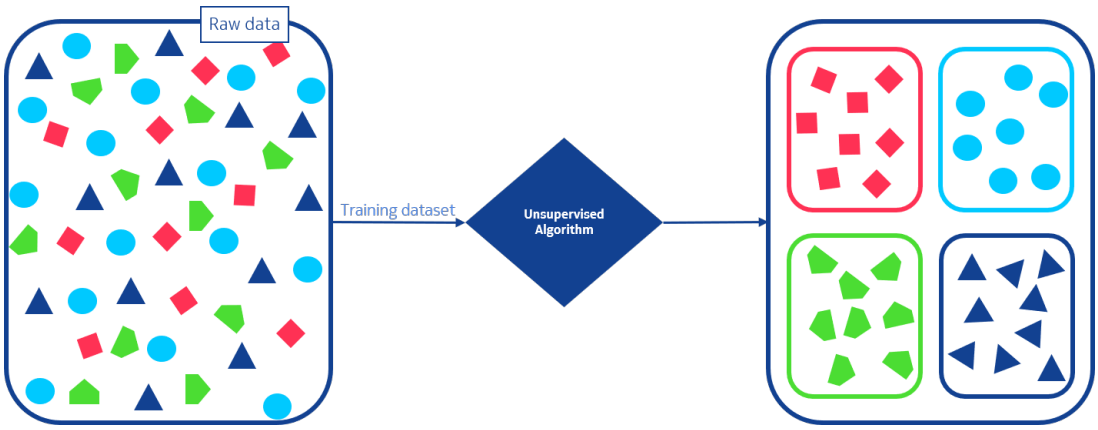

Figure 2.3 – Scheme of an unsupervised learning algorithm

1. Clustering: Gathering a set of inputs into divided groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.
2. Density Estimation: Looking for the inputs distribution in some space.
3. Dimensionality Reduction: Simplifying the inputs set by projecting them into a lower-dimensional space.

2.1.3 Reinforcement Learning

In cases of supervised learning, tuples $\{ (\boldsymbol{x}_i, \boldsymbol{y}_i) \mid (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{X} \times \mathbb{Y} , i \in [1, N] \}$ are available. However, the case of reinforcement learning we have unsupervised problems that only receive re-feeds or reinforcements (for example, win or lose). The supervised information $\boldsymbol{y}$ is replaced by information of the action/reaction type. The goal in reinforcement learning is to learn to map action situations to maximize a certain reward function. In these problems an agent learns by trial and error in a dynamic and uncertain environment. In each interaction, the agent receives as input a current status indicator and selects a certain action that maximizes a reinforcement or reward function in the long term (see figure 2.4).

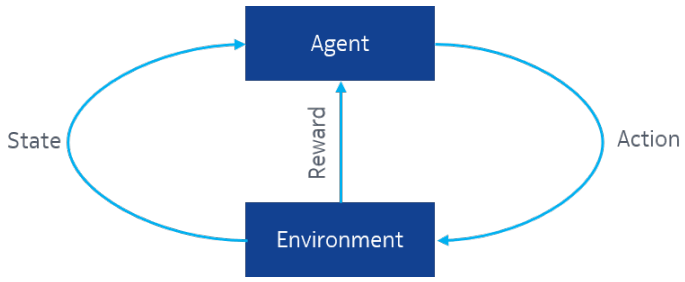

Figure 2.4 – Scheme of an Reinforcement Learning algorithm

2.1.4 More types

Actually, these three Learning types presents the main 3 basic types. However there are other types that are derived from these 3 (Supervised, unsupervised, Reinforcement Learning). For example, we can find the hybrid learning which is in general a combination between supervised and unsupervised Learning. We tried to summarize most of the other types in the table 3.3.

2.2 Machine Learning algorithm families

The idea of creating a "thinking" or "intelligent" machine is at least as old as modern computing, if not even older. Arthur Samuel first came up with the phrase “Machine Learning” in 1952. Few years after, the term "artificial intelligence" was born during the Dartmouth Workshop in 1956, which is widely considered to be the founding event of artificial intelligence as a field [49]. Nowadays, Machine Learning (ML) is an important aspect of modern business and research. It uses algorithms to assist computer systems in progressively improving their performance. Machine Learning algorithms automatically build a mathematical model using data samples (also known as training data) to make decisions without being specifically programmed to make those decisions. In this section, we will present some main families of ML algorithm which can be summarized with some example and main application fields in the figure 3.8.

2.2.1 Classical Machine Learning/ Shallow Learning

In literature, no verbatim reference has been found for "classical machine learning" expression, the common annotation is referred in short as "machine learning". However, with the apparition of the Deep Learning revolutionising concept in 2006, other machine learning algorithms have been called by experts as shallow learning algorithm or classical Machine learning. Technically, all machine learning families/methods mentioned in this chapter even the Deep Learning belongs indeed to the "machine learning" concept or field (see figure 3.9). Thus, for clarity and disambiguation reasons, it will be refereed in this report as "classical machine learning" for old algorithms (before Deep Learning) and not "machine learning".

Classical Machines Learning (CML) algorithms are often grouped by similarity in terms of their function (how they work). That's to say they are grouped according to their abilities to seek the structural patterns in the data in other words the way the inputs are mapped to the outputs [50], [51]. Actually, the pattern mapping data to each other can take multiple representations and structures that CML aims to look for in the model training building phase. Each one of these representations and structures dictates the kind of technique that can be used to infer that output structure from data. In fact, there are many different kinds of simple structure that datasets can exhibit. In one dataset, there might be a single attribute that does all the work and the others may be irrelevant or redundant. In another dataset, the attributes might contribute independently and equally to the final outcome. A third might have a simple logical structure, involving just a few attributes that can be captured by a decision tree. In a fourth, there may be a few independent rules that govern the assignment of instances to different classes. A fifth might exhibit dependencies among different subsets of attributes. A sixth might involve linear dependence among numeric attributes, where what matters is a weighted sum

of attribute values with appropriately chosen weights. In a seventh, classifications appropriate to particular regions of instance space might be governed by the distances between the instances themselves. And in an eighth, it might be that no class values are provided: the learning is unsupervised. Therefore, the CML Algorithm, more generally the ML algorithm, must be a chosen carefully. According to [51], [50] and [52], CML main methods are as follow (see figure 2.5):

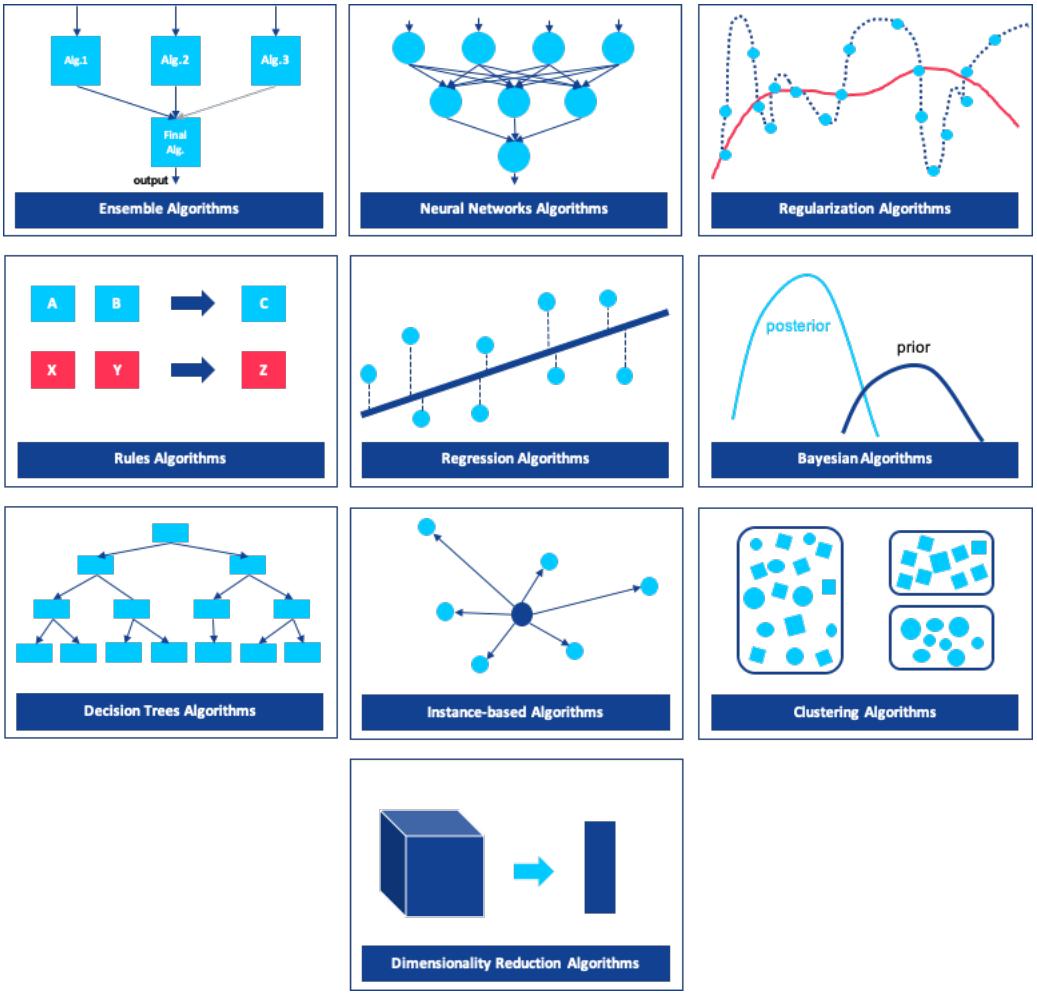

Figure 2.5 – Machine Learning Algorithms Modelling according to [50]

1. **Ensemble:** Ensemble algorithms in CML aim to combine the decisions/ prediction from multiple models to improve the overall performance. Ac-

tually, error causes in CML are due to the variance, noise and bias. Thus, inferring the final decision from many other models helps to minimize these error factors. These methods are designed to improve the stability and the accuracy of CML.

$\xrightarrow{\text{Eg.}}$ Boosting, AdaBoost, Bootstrapped Aggregation (Bagging), Weighted Average (Blending), Stacked Generalization (Stacking), Gradient Boosting Machines (GBM), Gradient Boosted Regression Trees (GBRT), Random Forest (the most commonly used algorithm)

2. **Neural Networks:** Neural Network concepts are inspired from biological neurons in the brain. Neural Networks experiments began as an attempt to mimic the architecture of the human brain to perform tasks that helps computer to be more intelligent. The main concept behind Neural Networks is a weighted graph composed of a neurons. These neurons are connected to each other in various patterns, to allow the output of some neurons to become the input of others. These algorithms are not recent, they were deployed since the mid of 20^{th} century. However, they quickly reached their limits since they are greedy in resources. With technological progress and the appearance of GPUs, Neural Networks are more and more exploited with a deeper graphs what gave birth to Deep Learning concepts (see figure 2.6). Deep-learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth (horizontally or the number of layer) and width (vertically or the number of neurons per layer) through which data must pass in a multistep process of pattern recognition. Neural Network and more precisely Deep Learning will be properly addressed in section 2.2.2 of this chapter.

   $\xrightarrow{\text{Eg.}}$ Perception, Back-propagation, Hopfield Network.

3. **Regularization:** Regularization algorithms or techniques are an exten-

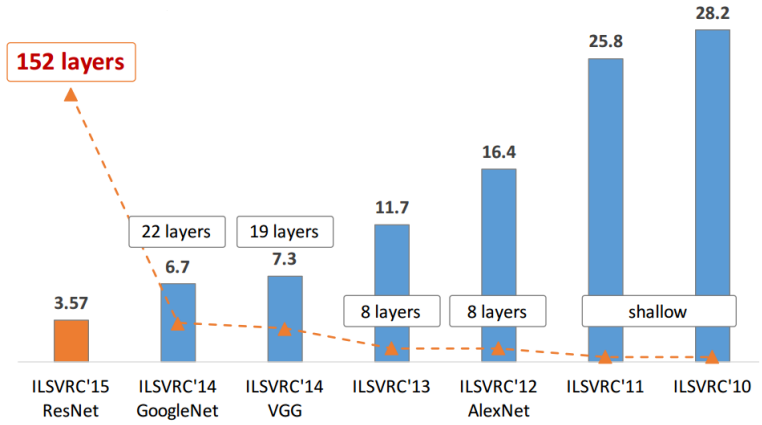

Figure 2.6 – Revolution of Neural Network Depth (number of layers) [53]: Eg. of image classification: Layers Number vs Model error (%)

sion added to other algorithm such as neural Networks or regression in order to attempt to penalize the model (or the overfitting) in a statistical way.

$\xrightarrow{\text{Eg.}}$ Ridge Regression, Least Absolute Shrinkage and Selection Operator (LASSO), Elastic Net, Least-Angle Regression (LARS).

4. **Rule System:** Association rule learning algorithms aim to find the relationships inside a given dataset.It identifies frequent if-then associations called association rules which consists of an antecedent (if) and a consequent (then). Eg. if milk and coffee then sugar.

   $\xrightarrow{\text{Eg.}}$ Cubist, One Rule (OneR), Zero Rule (ZeroR), Repeated Incremental Pruning to Produce Error Reduction (RIPPER), Apriori algorithm, Eclat algorithm.

5. **Regression:** Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model. It is usually used to predict a continuous value (quantitative variable). Predicting prices of a house given the features of house like size, price etc is one of the most known examples of Regression.

$\xrightarrow{\text{Eg.}}$ Ordinary Least Squares Regression (OLSR), Linear Regression, Logistic Regression, Stepwise Regression, Multivariate Adaptive Regression Splines (MARS), Locally Estimated Scatterplot Smoothing (LOESS)

6. **Bayesian:** Bayesian algorithms are a family of probabilistic classifiers that explicitly apply Bayes' theorem with strong naive independence assumptions between the features.

   $\xrightarrow{\text{Eg.}}$ Naive Bayes, Gaussian Naive Bayes, Multinomial Naive Bayes, Averaged One-Dependence Estimators (AODE), Bayesian Belief Network (BBN), Bayesian Network (BN).

7. **Decision Tree:** Decision Trees are a class of very powerful Machine Learning model cable of achieving high accuracy in many tasks while being highly interpretable. The main concept behind the decision trees is to split data continuously according to a certain parameter or value of features. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

   $\xrightarrow{\text{Eg.}}$ Classification and Regression Tree (CART), Iterative Dichotomiser 3 (ID3), C4.5 and C5.0 (different versions of a powerful approach), Chi-squared Automatic Interaction Detection (CHAID), Decision Stump, M5, Conditional Decision Trees.

8. **Instance-based learning:** Instance-based learning model is a decision problem with instances or examples of training data that are deemed important or required to the model. They are typically based on the distance or similarities inside the data set to determine which member of the training set is closest to an unknown test instance. Once the nearest training instance has been located, its class is predicted for the test instance. For this reason, instance-based methods are also called winner-

take-all methods and memory-based learning.

$\xrightarrow{\text{Eg.}}$ K-Nearest Neighbor (KNN), Learning Vector Quantization (LVQ), Self-Organizing Map (SOM), Locally Weighted Learning (LWL), Support Vector Machines (SVM).

9. **Clustering:** Clustering, like regression, describes the class of problem as well as the class of methods or algorithms. Clustering algorithms aim mainly to divide data to a number of clusters or groups according to the existing similarities in data instances. That is to say that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups.

   $\xrightarrow{\text{Eg.}}$ k-Means, k-Medians, Expectation Maximisation (EM), Hierarchical Clustering.

10. **Dimentionality Reduction:** Dimentionality Reduction, like regression and clustering, describes the class of problem as well as the class of methods or algorithms. Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, but in this case in order to summarize or describe data using less information.

    $\xrightarrow{\text{Eg.}}$ Principal Component Analysis (PCA), Principal Component Regression (PCR), Partial Least Squares Regression (PLSR), Sammon Mapping, Multidimensional Scaling (MDS), Projection Pursuit, Linear Discriminant Analysis (LDA), Mixture Discriminant Analysis (MDA), Quadratic Discriminant Analysis (QDA), Flexible Discriminant Analysis (FDA).

11. **Others:** There are many other algorithms that are not covered by this categorization like algorithm for specialty tasks in the process of machine learning, or specialized algorithms dedicated to some fields.

    $\xrightarrow{\text{Eg.}}$ **Processing Algorithms:** Feature selection algorithms, Accuracy evaluation algorithms, Performance measures, Optimization algorithms

$\xrightarrow{\text{Eg.}}$ **Specialized algorithms:** Computational intelligence (evolutionary algorithms, etc.), Computer Vision (CV), Natural Language Processing (NLP), Recommender Systems, Reinforcement Learning, Graphical Models, And more...

This is a useful grouping of CML algorithms since it gives an idea how the algorithm would deal with data 3.9, but it is not perfect. There are still algorithms that could just as easily fit into multiple categories like Learning Vector Quantization that is both a neural network inspired method and an instance-based method. There are also categories that have the same name that describe at the same time the Learning type and the class of algorithm such as Regression and Clustering.

Actually, CML outcomes is highly linked to the data distribution as well as the data nature and the data volume as well as the CML learning behavior vis-a-vis the data set. Indeed CML have always some conditions to fulfill for an optimal results which make data mining harder and more complicated regarding the data features at the disposal. Table 2.1 critics some methods according to [52].

2.2.2 From shallow (Classical Machine Learning) to Deep Learning

In order to develop intelligent systems able to approximately reproduce human capabilities and skills, we turned to neuromimetics (a system in which computational models methods apply underlying concepts of neural processes), that is to say the possibility of mimicking the brain, which allows man to reason, speak, calculate, compute and learn. This gives birth to two possible ways of mimicking:

- Mimicking cognitivism: which attempts to reproduce human reasoning

Method	Requirements	Problem
Decision Tree	Small set of possible values for each attribute	With a large data set, a decision Tree becomes large and illegible
Bayes	The attributes independence assumption is verified. The value normal distribution assumption for numerical attributes is verified.	It is a bit hard to verify the value normal distribution assumption.
Distance		Greedy on resources Long run time execution Difficult to pick the right distance/dissimilarity. Inability to deal with the noise
Neural Networks	Well set the architecture : Layer Numbers, neurons numbers, network topology	Look for a local optimum and a non global one.

Table 2.1 – Special requirements and problems with some algorithms of CML

and intelligence. It will give birth to the discipline of artificial intelligence.

- Mimicking connectionism: which attempts to reproduce the connection between neurons following the functioning of the nervous system.

Works on neuronal models dates back at least to the 1940s with Mc Culloch and Pitts [54] trying to model the brain cell network as closely as possible. They use the Hebb rule (*" cells that fire together, wire together"*) referring to the synaptic contacts between 2 cells when they are active simultaneously in the human brain. In 1958, Frank Rosenblatt [55] presents the perceptron. This is the first model for which a learning process is defined. While the perceptron operates in a binary mode, Widrow and Hoff [56] propose in 1960 a version that reacts linearly to the input signals. This version is called ADALINE for

ADAptative LINear Element.

The beginning of the first decade of the 21st century turned out to be a turning point in the history of ML by the apparition of Deep Learning, and this is explained by the three simultaneous trends, which together gave a noticeable synergetic effect. The first is Big Data. There is alot of abundant data tha can be exploited by the curiosity of scientists. The second is the decrease in the cost of parallel computing and memory. This trend was discovered in 2004 when Google unveiled its MapReduce technology, followed by its open analogue Hadoop (2006), and together they gave the opportunity to distribute the processing of huge amounts of data between simple processors. At the same time, Nvidia made a breakthrough in the GPU market for in the gaming purposes, then it turned out that it can be used for machine learning purposes. In these condition, A leading trio has emerged: Yann Lecun, Yoshua Bengeio and Geoffrey Hinton [32]. Supported by Google and Facebook, they have advanced research in the field and they have shown a great performance of Deep Learning in resolving complex problems that was unsolved so far. Statistically, Deep Learning methods are beating out shallow and classical machine learning approaches in every single metric [57].

The key concept of Deep Learning is the "neuron" which is the constructive unit of a deep neural network. So mathematically, an artificial neuron is simple computational unit which will make a particularly computation function based on other units which is connected to. More formally a neuron is characterized by (see figure 2.7):

- The neurons set to which it is connected. This set constitutes its input neighborhood usually modelized with a vector noted $\boldsymbol{x}$, $\boldsymbol{x} = \{x_0, ..., x_n\}$ where n is the number of neurons to which it is connected.
- Weights $\boldsymbol{w} = w_0, ..., w_n$ a weight for every connection.
- Biais b

- A preactivation score z that is computed by adding the bias added to the sum of the connected neurons multiplied by their weights as follow: $z = \sigma(x) = b + \sum_i w_i x_i = b + \boldsymbol{wx}$.
- Activation function g that takes as input the preactivation score $g(z) = g(\sigma(x)) = b + \sum_i w_i x_i = b + \boldsymbol{wx}$
- Output of activation function : $a = g(z) = g(\sigma(x)) = b + \sum_i w_i x_i = b + \boldsymbol{wx})$

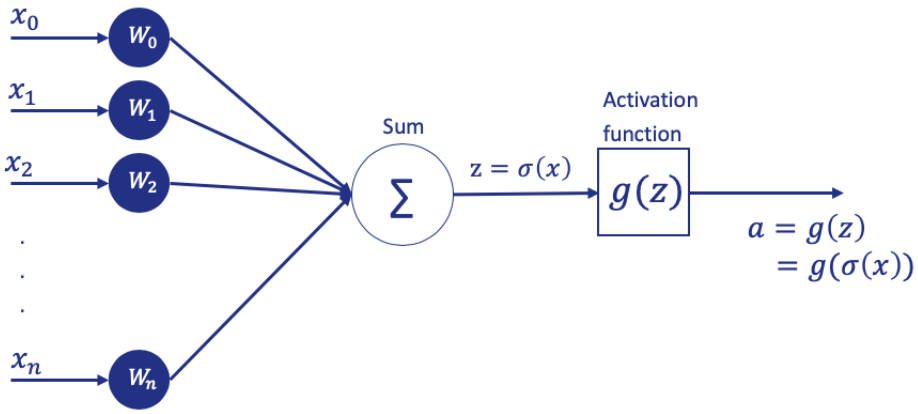

Figure 2.7 – Neuron characterization: connection, weights, bias, activation function

Thus, a neural network is a structure made up of stacked multilayer neural. There are many topologies of the network connections: Hopfield, recurrent, ect. However we are rather interested in the feedforward neural network topology. These models are called feedforward because of the information flows through the function they approximate. This flow starts from $\boldsymbol{x}$ (input data vector) then passes through the intermediate computations of the hidden layers, and finally to the output layer describing the output $\boldsymbol{y}$. There are no feedback connections in which outputs of the model are fed back into itself. Thus we can consider the feedforward Neural Network (FNN) as a directed acyclic graph describing how the functions (layers) are composed together. For example, Let's consider a FNN composed of 3 layers so we might have three functions f^3,f^2, and f^1 connected

in a chain, to form $f(x) = f^3(f^2(f^1(x)))$ where f^1 is the first layer of the FNN and f^2 is the second layer and f^l is the l^{th} layer of the FNN. The overall length l of the graph formed by the FNN gives the depth of the model. Thus, the term Deep Learning came from this terminology. The first layer receives raw data as input, it is called the input layer. The intermediate layers, called hidden layers, receive as inputs the outputs of the neuron of the previous layer. The last layer, being the output layer, returns the expected result of the algorithm. Each hidden layer f^l of the network is typically vector valued composed of neurons [1] . The number of these neurons will define the width of the FNN. The output of each neuron is calculated as follows (see figure 2.8):

$$a_j^l = g(z), \text{where } z = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$$

Where:

— a_j^l: the output of the j^{th} neuron in the layer l.

— b_j^l: the bias of the j^{th} neuron in the layer l.

— w_{jk}^l: the weight of the connection of the k^{th} neuron in the layer $l-1$ to the j^{th} in the layer l.

Activation Function

In literature there are many potential choices for the activation function of a neuron g. We can categorize them according to their types: Identity function, Binary step function, Bipolar step function, Sigmoidal function, Binary sigmoidal function, Bipolar sigmoidal function, Ramp function. The activation function choice is made according to the properties and the nature of the data as well as the target task we aim to resolve. Knowing our data property as well as the activation function property can drastically help the Neural Network to convergence. We enumerate bellow some activation functions (see figure 2.9)

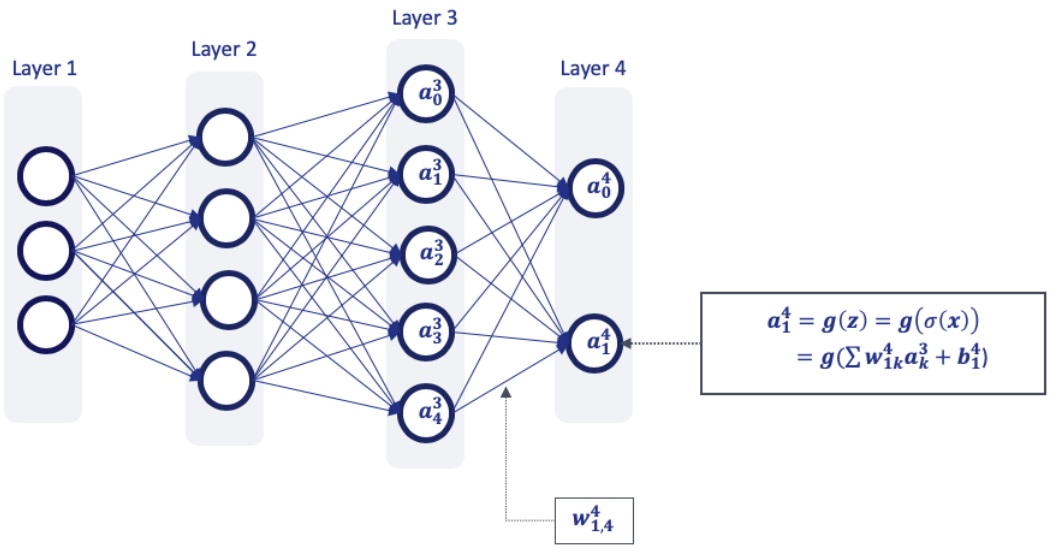

Figure 2.8 – Feedforward Neural Network (FNN)

as well as their properties.

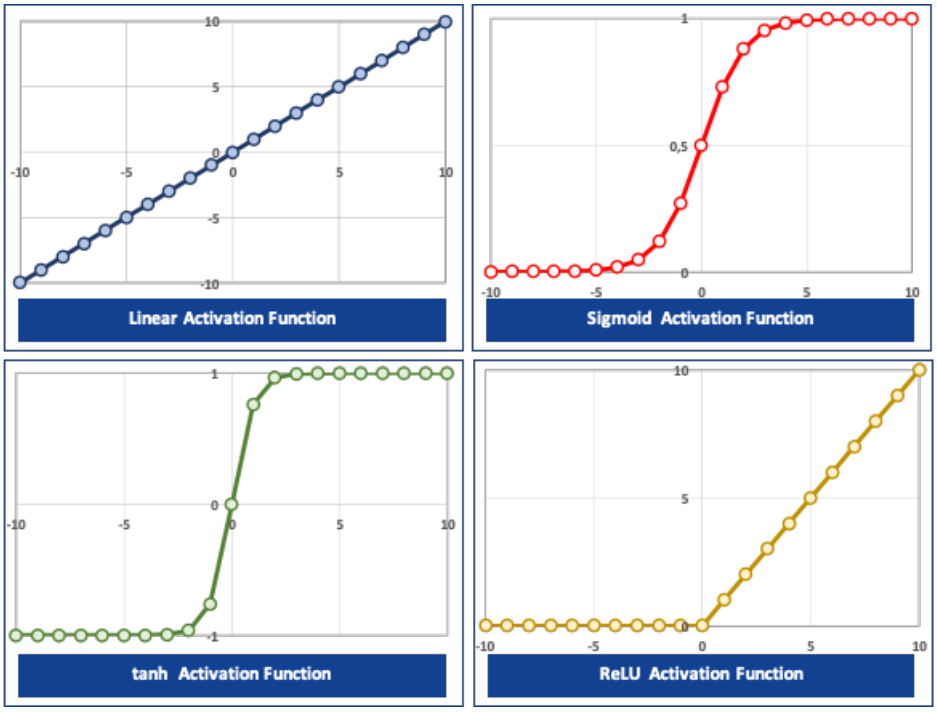

Figure 2.9 – Some Examples of Activation Functions: Linear, Sigmoid, tanh, ReLU

1. **Binary function:** A threshold binary function that switches the output between two values either 0 or 1 if the activation exceed a certain threshold value k.
$$\begin{cases} g(z) = 0 \quad if \quad z < k, \quad \forall k \in \mathbb{R} \\ g(z) = 1 \quad if \quad z > k, \quad \forall k \in \mathbb{R} \end{cases}$$

2. **Linear function (or called also identity):** the simplest choice is to make g as a linear function. So in this case we will have $g(z) = z$ (see figure 2.9). However, this choice does not perform any squashing of the input, that's to say it takes the input and reproduces it so it won't be a bounded activation function which makes it not really an interesting choice, since it doesn't introduce any non linearity during the computation of the neuron which may be not useful to reproduce a real world complexity tasks.

3. **Sigmoid function:** A more interesting choice which takes:
$$g(z) = \frac{1}{1 + \exp^{-z}}$$
By plotting this function (see figure 2.9) we can easily note that neuron activation is squashed between $[0, 1]$ (all output values are between $[0, 1]$): the bigger the preactivation is, the more the output tends towards 1 and the smaller the preactivation is (when it goes to $-\infty$) the more the output tends towards 0 instead. So this activation function is always positive (always between $[0, 1]$). It's also bounded, the result can't be smaller than 0 or bigger than 1. And it is a strictly increasing function : the bigger the preactivation is, the higher the output is.

4. **Hyperbolic tangent function:** Hyperbolic tangent ($\tanh$) activation function is another popular function (like the previous one the sigmoid activation function). It is likely more complicated. It also involves some ex-

ponentials:

$$g(z) = \tanh(z) = \frac{\exp^z - \exp^{-z}}{\exp^z + \exp^{-z}} = \frac{\exp^{2Z} - 1}{\exp^{2z} + 1}$$

By plotting this function (see figure 2.9) we can easily note that neuron activation is squashed between $[-1, 1]$ (all output values are between $[-1, 1]$). So this function output can be either positive or negative. It's also bounded and strictly increasing (just like the sigmoid function).

5. **Rectified Linear activation function (ReLU):** A popular choice, it's less complicated than the sigmoid and the hyperbolic tangent activation function. Simply, it's the maximum between 0 and the preactivation value z:

$$g(z) = ReLU(z) = max(0, z)$$

By plotting this function (see figure 2.9) we can easily note that it's always non negative (only bounded below by 0 and not upper bonded indeed the greater the preactivation is the higher the output will be). It's a monotonically increasing function. In practice this functions tends to give neurons that have sparse activities that's to say it tends to get neurons that are often exactly zero. Which is not the case for the sigmoid or tanh activation function since both of them need to have a preactivation with exactly a particular value.

6. **Softmax activation functions:** A final popular choice, mainly used to handle multiple classes. Simply, it normalizes the outputs for each class between 0 and 1, and divides by their sum, giving the probability of the input value being in a specific class

$$g(z)_j = \frac{\exp^{z_j}}{\sum_k \exp^{z_k}} \forall j \in \{1..k\} \text{ with } k \text{ the number of classes}$$

Typically Softmax is used only for the output layer, for neural networks

that need to classify inputs into multiple categories or classes.

7. **Other activation functions:** There are many other activation function like the swish function a self-gated activation function discovered by Google researchers [58] and performs better than ReLU with a similar level of computational efficiency. There are also Leaky ReLU, and Parametric ReLU that are both inspired from ReLU function aiming to deal with the ReLU biggest problem (The Dying ReLU problem when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn). Note that we can also customize our activation function according to the needed task or according to our data specificity.

Learning Concept & Cost function

Let's assume that our main goal for using the neuronal network is to come by the end to detect if a given image is either a dog or a cat. So to properly work, at first the neural network must receive the image of the dog or the cat which will be modelized as an input vector that we denoted x (more precisely x^l where l is the layer's index or the layer's number). Once it receives the initial value of the vector x^0 (in our case the image), the network proceeds with a set of computations to produce an output y which will be the response of the network (either the image is a dog or a cat). This final decision is highly dependent on the activation functions of the neurons, as well as the weights and the bias. Obviously, we would like that the neuronal network comes to fulfill this particular task by providing the correct answer (denoted here $\hat{y}$ to designate desired response). It is this desired answer $\hat{y}$,that the network should have given, which makes it possible to know if it made a mistake compared to the answer y.

It should be remembered that in order to carry out a learning process, we must have a set of examples, also known as the learning base or learning dataset or training dataset. This dataset have to be statistically representative of the problem to be solved. In general, during the learning phase in a neuronal network, we aim to minimize the error between the output $\boldsymbol{y}$ and the desired output $\hat{\boldsymbol{y}}$. It is a cost function which finds a set of weights $\boldsymbol{W}$ and the bias $\boldsymbol{b}$ approximating the calculated outputs $\boldsymbol{y}$ of the desired outputs $\hat{\boldsymbol{y}}$. Let θ be the set of weights $\boldsymbol{W}$ and the bias $\boldsymbol{b}$ approximating the calculated outputs. Thus, to achieve successfully the learning phase, a distance measure was set between the desired answer $\hat{\boldsymbol{y}}$ and the calculated output $\boldsymbol{y}$. This measure corresponds to the error committed by the network which is due to the current values of θ. The simplest function of this distance is the least squares distance, which measures the quadratic error between $\boldsymbol{y}$ ans $\hat{\boldsymbol{y}}$. We note large J this function which is called cost function or objective function. This error is therefore the sum of the errors on each of the examples of the learning base.

$$J(\theta) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\| = \sum_i (\hat{\boldsymbol{y}} - \boldsymbol{y})^2$$

The training process based on minimizing this average training error is known as empirical risk minimization. Let P be the some performance measure minimizing the error between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$. Typically, the cost function can be written as an average over the training set [1], such as:

$$J(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim P}[L(f(\boldsymbol{x},\theta),\boldsymbol{y})] = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i,\theta_i),y_i)$$

where:

— L is the loss function that computes the distance between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$. Some examples of the used loss functions are presented in table 2.2 [1].

1. https://keras.io/api/losses/

— $f(\boldsymbol{x}, \theta)$ is the predicted output when the input is $\boldsymbol{x}$ ($f(\boldsymbol{x}, \theta) = \hat{\boldsymbol{y}}$)

— N the number of the training examples.

The goal of the DL algorithm is to reduce the expected generalization error given by this equation. To find the best parameters θ that minimize the best the function J, a gradient descent method is used. This requires calculating the partial derivatives of J with respect to each parameters θ. This calculation is made possible thanks to the algorithm of the backpropagation of the gradient that we will describe in the next section.

Classification		Regression	
Function	Formula	Function	Formula
Binary Crossentropy	$-(\boldsymbol{y} log(\hat{\boldsymbol{y}}) + (1 - \boldsymbol{y}) log(1 - \hat{\boldsymbol{y}}))$	Mean Square Error (MSE)	$(\hat{\boldsymbol{y}} - \boldsymbol{y})^2$
Categorical Crossentropy	$-(\sum_{j=0}^{M} (\boldsymbol{y}_j log(\hat{\boldsymbol{y}}_j)))$, M the class number	Mean Absolute Error (MAE)	$\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|$
Kullback Leibler Divergence Loss	noted $D_{\mathrm{KL}}(P \| Q)$, $\boldsymbol{y} log(\frac{\boldsymbol{y}}{\hat{\boldsymbol{y}}})$	Mean Squared Logarithmic Error Loss	$(log(\boldsymbol{y} + 1) - log(\hat{\boldsymbol{y}} + 1))^2$
Squared Hinge Loss	$max(0, 1 - \boldsymbol{y}, \hat{\boldsymbol{y}})^2$	Poisson	$\hat{\boldsymbol{y}} - \boldsymbol{y} log(\hat{\boldsymbol{y}})$

Table 2.2 – The most known loss functions

Backpropagation algorithm: Stochastic Gradient Descent (SGD)

In order to minimize $J(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim P}$ we need to update the weights related to each neuron. One common approach to do this weight update is via the use of a a backpropagation approach like the stochastic gradient descent algorithm [46]. SGD is a stochastic approximation of the gradient descent method for minimizing an objective function that is written as the sum of differntiable function [59]. For a given objective function, we can obtain the gradient with

respect to the model parameters using calculus and applying the chain rule. To obtain the gradients with respect to the parameter set θ we compute partial deviates of $J(\theta)$ with respect to the parameters θ_j of each layer $j = \{0, .., K-1\}$ whith K is the total number of layers. The chain rule allows us to determine the partial derivatives as:

$$\frac{\partial J}{\partial \theta_{k-1}} = \frac{\partial L}{\partial \theta_{k-1}} = \frac{\partial L}{\partial f_k} . \frac{\partial f_k}{\partial \theta_{k-1}}$$
$$\frac{\partial J}{\partial \theta_{k-2}} = \frac{\partial L}{\partial f_k} . \frac{\partial f_k}{\partial f_{k-1}} . \frac{\partial f_{k-1}}{\partial \theta_{k-2}}$$

$$\vdots$$

$$\frac{\partial J}{\partial \theta_j} = \frac{\partial L}{\partial f_k} . \frac{\partial f_k}{\partial f_{k-1}} \cdots \frac{\partial f_{j+2}}{\partial f_{j+1}} . \frac{\partial f_{j+1}}{\partial \theta_j}$$

The first order derivative $\partial J / \partial \theta_j$ enables whether the error J is increasing or decreasing when the values of the parameters (weights and bias) is equal to θ_j. Thus, the derivative measures how much the parameter θ needs to change (in positive or negative direction) to minimize J (for further readings see [1], [59] and [60]).

To sum up, a FNN steps:

1. Build the Neuronal Network and intialize it.
2. Repeat until a fixed iteration number (epoch):
   (a) Calculate the forward phase: $\forall(\boldsymbol{x}_i, \boldsymbol{y}_i)$ compute $\hat{\boldsymbol{y}}_i$.
   (b) Compute the error between $\hat{\boldsymbol{y}}_i$ and $\boldsymbol{y}_i$
   (c) Calculate the backpropagate phase and adjust θ

Over-fitting & Under-fitting

CML as well as DL mainly aim to find a function $\hat{f}$ based on input data $\boldsymbol{X}$ that approximate $\boldsymbol{Y} = f(\boldsymbol{X})$. Statistically, a fit refers to how well an ML model approximate a target function. We mean by a good approximation if the ML model generalizes any new input data from the problem domain (same data distribution or same $P(\boldsymbol{Y}|\boldsymbol{X})$) in a proper way. This generalization allows to make predictions in the future data, that data model has never seen. If the generalization is not reached, we are then facing either an underfitting or an overfitting that are modelized in the figure 2.10

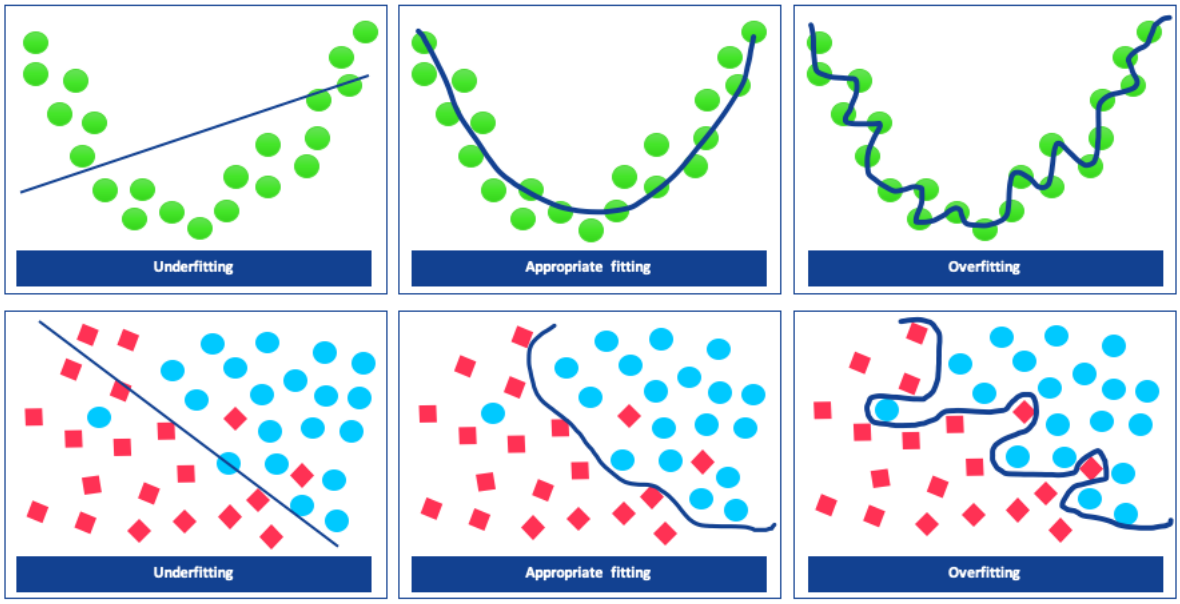

Figure 2.10 – Examples of Underfitting and Overfitting in Machine Learning: Regression and classification

— Underfitting: when the ML model can't capture the underlying trend of the data.It usually happens if the ML model is too simple compared to the task we want to resolve. Techniques to reduce underfitting (that we will detail in chapter 3):

  - Increase model complexity
  - Increase number of features, performing feature engineering
  - Remove noise from the data
  - Increase the number of epochs or increase the duration of training to get better results
- overfitting: ML model learns data by heart. Overfitting is often a result of an excessively complicated model. Techniques to reduce overfitting (that we will detail in chapter 3):
  - Increase training data.
  - Reduce model complexity.
  - Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training)
  - Ridge Regularization and Lasso Regularization
  - Use dropout for neural networks to tackle overfitting.

Understanding the issues of overfitting and underfitting takes us right back to the bias and variance dilemma. The name bias-variance dilemma comes from two terms in statistics: bias and variance. Let $\mathbb{E}$ be the expectation. The ML error can be decomposed into bias and variance components following this formula:

$$\mathbb{E}[(f - \hat{f})^2] = \mathbb{E}[(y - \hat{y})^2] \quad = Biais(\hat{f})^2 + Var(\hat{f})$$

The bias term measures the error of estimations, and the variance term describes how much the estimation $\hat{y}$ moves around its mean. Thus underfitting occurs if the model or algorithm shows low variance but high bias. As for the overfitting, it occurs when the ML model shows low bias but high variance. Both overfitting and underfitting lead to poor predictions on new datasets. The best ML model that generalizes and approximate better the data is a tradeoff

between the bias and the variance.

2.3 Conclusion

In this chapter, we have introduced ML basics for both CML (or shallow learning) and DL with some definitions and generalities. We have presented ML learning categories (supervised learning, unsupervised learning and reinforcement learning) as well as ML algorithm families. We have also presented DL basics: some history, definition, generalities, and its operation mode.

Chapter 3

Machine Learning Workflow Toward User Behavior Characterisation

Recent technological breakthroughs have extended the mobile phones' features, functions and capabilities, which are now used for more than just communicating or affording applications. Recently, mobile devices are being utilized to know the consuming habits of individuals and communities. For example, in [61] authors suggests that understanding how mobile consumers use smartphones for shopping is important in developing digital shopping platforms fulfilling consumers' expectations. In [62] investigates news consumption on mobile devices with the goal of identifying where mobile devices fit into people's media repertoires and how consumption patterns on them are different from those on other platforms. As for authors in [63] authors study applied the mobile habit consumption to explain the intentions of Hong Kong consumers to adopt mobile TV and their interests in its content. Examples are abundant in the literature where the analysis of the behavior of the mobile user is made in order to optimize some services or propose customized services according to the habits.

Considering the fact that users carry their smartphone devices with them daily (24h/24h and 7d/7d), smartphone devices allow collecting data, mea-

sured by phones, in various situations experienced by their owners. According to some analytics platform [1], the total number of global smartphone users in the world, in 2019, is around 3.3 billions, where around 521 millions of these users are located in Europe (Figure 3.1). With the high volumes of smartphones deployed worldwide, data measured directly inside mobile phones consists of very huge amount of representative data.

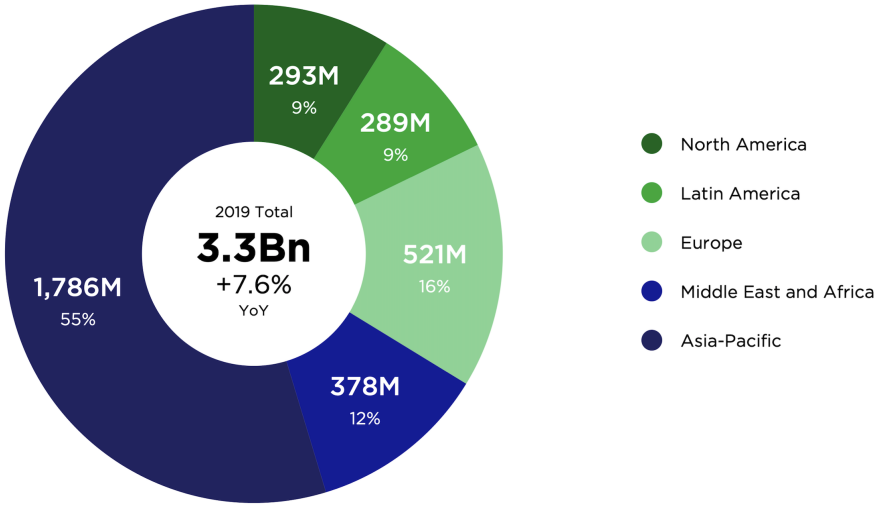

Figure 3.1 – Active smart phone users in the world and per region according to 2019 Stats (Study carried out by "newzoo" Analytics Platform).

Our goal is to extract knowledge of user behavior based on the real time processing of crowd-sourced data and considering a model of two QoE-influencing attributes of contextual information. We aim to infer user's environment and speed range with ML algorithms which model is created via the learning process (called also training process). The main goal of learning is to create an accurate model that correctly provides information most of the time. However, in order to train a model correctly, there is some methodology to respect. There is also a requirement to use relevant data depending on what we want to infer.

This chapter describes the methodology used to design an efficient user pro-

1. https://newzoo.com/key-numbers/

filing system using ML. It is organized as follows. **Sec. 3.1** presents the general methodology defined in the state of the art (SoA) to build an ML solution with good performances and details how it is projected in our user behavior problematic. In section **Sec. 3.2** we try to project the user behavior profiling into the ML workflow described in the previous section. In **Sec. 3.3**, we describe our dataset and present how we collected such highly representative data. Finally conclusion is presented in **Sec. 3.4**.

3.1 Methodology in 5 steps

ML is the solution towards automating tasks by consuming and understanding data smartly. Thus, achieving good results using ML algorithms depends a lot on data among other factors. Bur, once we have this data, how does it really work under the hood? In section 4.5 of [64] the author outlines a universal workflow of machine learning, which he describes as a blueprint for solving machine learning problems. This workflow is composed of 7 steps:

1. Defining the problem and assembling a dataset
2. Choosing a measure of success,
3. Deciding on an evaluation protocol,
4. Preparing your data,
5. Developing a model that does better than a baseline,
6. Scaling up: developing a model that overfits
7. Regularizing your model and tuning your parameters.

Such workflow, is very high level, where we are assuming from the first trial the model is good and trough the steps we are transforming the model from good to great model that will be scaled back and then generalized. Another workflow, a bit different from the previous workflow, presented by Google for

AI training sessions [65] seems more concerned with going from zero to good. This workflow proposes also 7 other steps:

1. Data collection,
2. Data preparation,
3. Choose model,
4. Train model,
5. Evaluate model,
6. Parameter tuning,
7. Make prediction.

In the description of the two workflows, they agree and focus together on particular points (like hyperparameter tuning, data collection, etc). However, they do not agree in the order in which the steps are linked in the overall functioning of the workflow. Mapping [64] and [65] to the classical workflow presented in [52] we propose a 5-step process that we have followed for our investigation and which are as follows (see also figure 3.2):

1. Identifying and analyzing needs.
2. Generating / collecting or retrieving high quality and representative data.
3. Analyzing and reducing data size.
4. Identifying appropriate methods as well as appropriate algorithms.
5. Validating the method's performance. If this step is reached and the ML algorithm's performance is below expectation, then there are a few retro-back options to follow in order to boost the performances.
   (a) If performance is not satisfactory, when we arrive to step 5, in that case we go back to step 4 and either try to adjust the hyperparameter of the algorithm or to change the algorithm itself.

(b) If we have already looped more than a handful of times between step 5 and step 4, but the performance is still far from our expectation, then we return to step 2 and try to generate more data.

(c) If even with more data, the performance is still not satisfactory then we return to step 1.

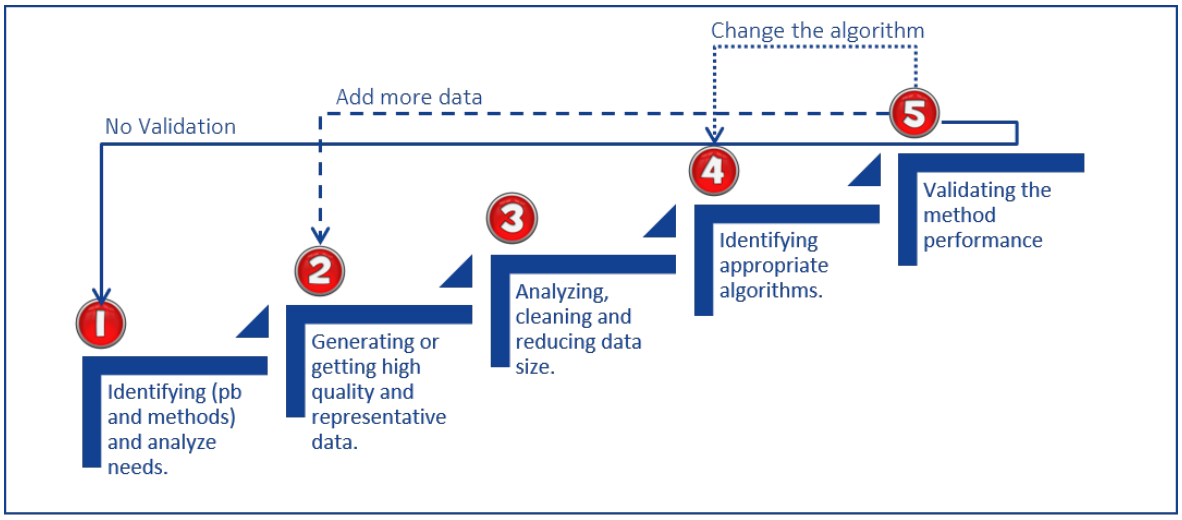

Figure 3.2 – Machine Learning in 5 main Steps

Now let's see these steps in more detail.

1) First, we begin by properly identifying and analyzing the needs of the problem, which we want to solve using ML. In other words, that means to fix the goal which we want to achieve as well as the expected output. Then, we define the nature of the learning task: either regression or classification or could be some other. After that, we define a success criteria that we want to reach. Usually, in ML this criteria is a performance score that measures how well the algorithm is performing. Thus, in this first step, we try to answer the question: "What is it that we want to find out? How will we reach the success criteria which we set?". Once this is fixed, one of the most important challenges in this first step, is to find out what are the inputs or features and the expected

outputs. In order to define these inputs as well as the outputs, first a set of questions must be answered:

. What is the main objective? What are we trying to predict or detect?
. What is the target output? What is its nature? (qualitative/ quantitative/ etc)?
. What can be correlated to such output?
. What is the most relevant input data or features to be considered? Is it feasible to have such an input? Does it describe the task?
. What kind of problem are we facing? Binary classification? Clustering?
. What is the current nature of the target features?
. How will the target feature be measured?
. How can we access the output? Is it possible to collect the output data, used for training, at the same time while collecting the input data?

It is crucial to keep in mind that ML algorithms are generally used to memorize patterns that are present in the training data. Thus, we can only learn what we have seen before. In other words when we are using ML approach, we are making the assumption that the future will mirror the past.

2) Thus, there is the need of the second step which is about data collection. We can consider the data collection as the first real and the most important step towards deployment of a ML process. It is a very critical and a sensitive step that will affect the quality of the model. The more and better quality data that we can get, the better our model will perform. In fact, data is a cornerstone for any ML process. If either the quality or quantity of data is not enough, then it can cause big issues and poor performances. Once our available data is informative enough to learn the relationship between the inputs and the outputs, we can move to the third step dealing with data preparation or data processing.

3) During this step, we aim to transform unstructured to structured data

which is ready for the training step. Raw data alone is not very useful. The data needs to be prepared before starting the next step. As it happens, collected data and especially real data is messy, noisy and has anomalies requiring cleaning. There is no perfect recipe or process to follow for cleaning, since it depends on data anomalies. Generally, data cleaning covers removing duplicates, correcting errors, dealing with missing values, normalization, data type conversions, removing outliers (Data points that differ significantly from other observations), etc. The cleaning process is not the only sub-step of data processing (see figure 3.3). Sometimes the collected data is not enough so that other forms of adjusting and manipulation are needed. Feature engineering can be seen as a kind of such adjusting and manipulation. Through it we make the appropriate transformations of some features to make them more significant or creating new input features from the existing ones. In general, data cleaning can be considered as the process of reduction and the feature engineering as a process of addition. Usually, such transformation, data manipulation and cleaning is conducted after analysis and visualization of features' behavior, correlation, nature, distribution, etc. Indeed, visualizing data helps a lot to detect relevant relationships between variables or class imbalances (which presents a big issue to handle), or perform other exploratory analysis. Once cleaning and feature engineering are completed, we generally randomize data, and we split it in 2 sub-datasets (i) training and (ii) testing. As we can see in figure 3.3, from the step 1 of "problem and needs identifying" to the step 3 of "data processing", we are dealing only with the data flow in the ML process.

4) and 5) The next step in our workflow is choosing an appropriate algorithm. As we explained, in chapter 3, precisely section 2.2, there are many algorithm families and many algorithms in each family. The choice of an algorithm depends on the task, the dataset and the needs. It is often valuable to compare some algorithms and pick the most suitable choice. For the chosen

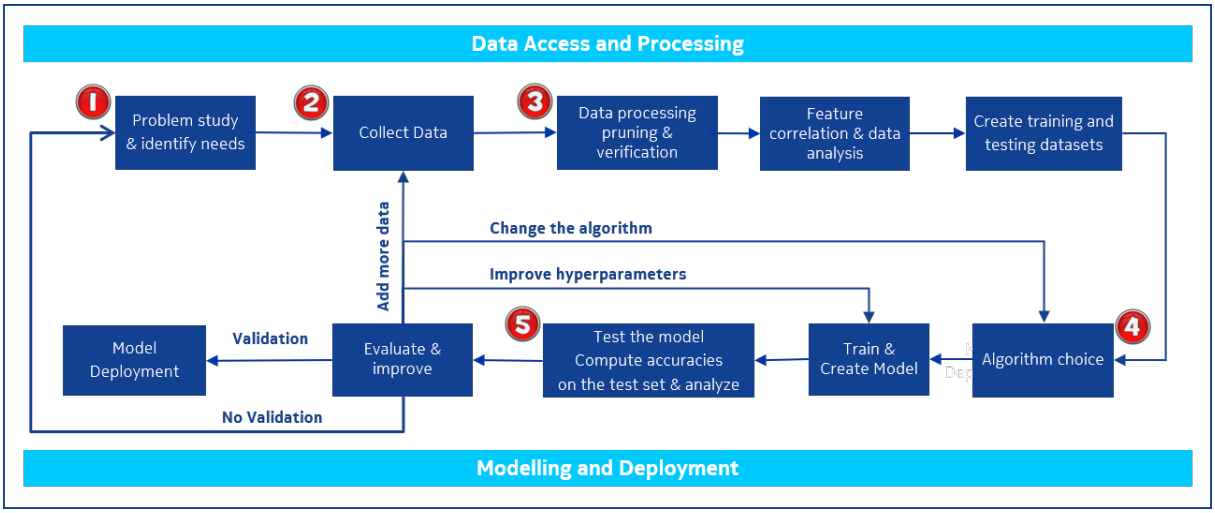

Figure 3.3 – Machine Learning Steps: Detailed Process

algorithm, we proceed with the training phase, where the algorithm tries to learn the pattern in the dataset. Once training is complete, the performance of the algorithm is evaluated according to the chosen success criteria on the testing set. This evaluation on the testing set allows us to test our model when facing new data which was never shown to the model while training. Here, two cases can arrive after this evaluation:

1. **Case 1: the success criteria is achieved** In this case, we assume that the model is good enough to be deployed and put into production.

2. **Case 2: the success criteria is not achieved** Rare are the cases where the performance evaluation achieves the success criteria from the first try itself. Usually, after a first training, we need to adjust the hyperparameters of the learnt model in order to optimise the overall performance. Each model has a variety of parameters that change how it makes decisions. We can adjust these and compare the chosen evaluation metrics of the different variants to find the most accurate model. In case we continue looping between training and adjusting the model hyperparameters

and a significant time is spent without any progress in the overall performances: it is necessary to ask the question "Was the right ML algorithm chosen for this task?" or "Is it a data issue?". In this case we try to change the algorithm and evaluate the performances. If it is still below expectation, then it is absolutely necessary to review the data flow: either the collection phase or the processing phase.

This process is a fairly standard process for evaluating an ML approach. However, in practice it can be done differently. These steps may not be linear. For example, feature selection or metric selection for performance evaluation can also be done after data analysis. This is because the nature of data or its distribution sometimes imposes some other choices rather than the first one. Also, in some cases data analysis and data relationship understanding are more important than knowing the impact of some features and the decorrelation of others. The important part is to carefully analyze each step in order to find the best model to deploy.

3.2 Methodology for user behavior prediction

In this section, we intend to apply the same methodology described in the previous section by projecting our use case of user behavior inference into the ML process workflow. It should be noted that we are not really detailing the user behavior detection according to this workflow, but rather fixing some basic points adopted to solve the challenge during the work. We recall that the objective is to answer the following questions: 1) where is the user while experiencing a mobile service? Is user indoor or outdoor? and 2) How is he consuming the mobile service? At which mobility speed range?. In this thesis, we investigate these questions one by one, in different chapters, respectively in chapter 4 and 5, then the questions simultaneously in chapter 6.

3.2.1 Identifying the problem and analyzing the needs

Inputs, Outputs and Learning task

Our answer to the question "What do we want to find out?" is that we want to detect simultaneously the user mobility or speed range as well as user environment while consuming a service. Therefore, the simultaneous user speed range and user environment detection are our target outputs. As we can notice, both speed-range and user environment are discrete variables. Thus, we are dealing with either a classification or clustering task. While collecting data, we collected the labels or outputs required so we are dealing with a classification task. An additional constraint is imposed to the data we select. Inputs used to achieve our goal must be known by the network and shall be correlated to the user environment and the user mobility speed range. This is important because we are interested in detection from the network side.

Success criteria and metrics

For the performance evaluation, we will use 2 classical metrics which are the most commonly used metrics in classification tasks [66], [67] Accuracy and F1 score. F1 score is more adapted to our use case, especially in the case of imbalanced data.

— Accuracy (ACC): It is one of the most used metrics to evaluate the performance of a ML model. It is presented according to the following formula:

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where:

. TP: True Positive: True positives are data points classified by the model as positive that are actually positive (meaning they are correct).

. TN: True Negative: True Negatives are data points classified by the model as negative that are actually negative (meaning they are correct).

. FP: False Positive: False positives are data points incorrectly classified by the model as positive that are actually negative (incorrect).

. Fn: False Negative: False negatives are data points incorrectly classified by the model as negative that are actually positive (incorrect).

Accuracy refers to the closeness of a prediction model (model classification) to the ground truth (the real outputs). However, accuracy can be misleading for imbalanced data sets. For example, for binary classification on data with 95% of class 1 and 5% class 2 values, classifying all values as class 1 gives 95% accuracy score. In this case, the model is satisfying the needs (since 95% is a good accuracy score), however it is not precise at all since is not detecting the class 2 samples at all.

— F1-score: A weighted score of precision (Pr) and recall (Rc) according to the following formula

$$F1 - score = 2.\frac{Pr.Rc}{Pr + Rc}$$

where

$$Pr = \frac{TP}{TP + FP} \text{ And } Rc = \frac{TP}{TP + FN}$$

Precision metric shows how precise the model is in terms of predicted positive results, or how many of them are actually positive. It gives an answer to the question "What proportion of positive identifications was actually correct?".

The recall metric quantifies how well the model is able to capture sam-

ples predicted as positive. Recall is therefore the metric to use in order to to select the best model when false negatives are high. It gives an answer to the question "What proportion of actual positives was identified correctly?". F1 score is a single metric that combines recall and precision using the harmonic mean. There are other metrics for combining precision and recall, such as the Geometric Mean of precision and recall.

We fixed our criteria of success to 95% of F1-score. Thus, we assume an error margin of 5%. This is inspired from network dimensioning requirement. Indeed, for mobile networks dimensioning, an error up to 5% is qualified as an admissible error rate. A further investigation has to be done to guess the relevant minimal value. But, it is out of scope of this thesis.

3.2.2 Data Processing

The flow of data collection process, mainly in the case of real data (in a crowd-sourced mode), is often done without a strict control, which leads to some noise and problems in the dataset like out-of-range values, impossible data combinations, missing values, etc. ML algorithms trained on data that has not been carefully processed, for such problems, can produce misleading results. As explained in next section 3.3, data processing includes: normalization, data cleaning, transformation, extraction and selection of variables, feature engineering, etc. The details of data analysis and processing will be discussed in the chapters 4 (User environment detection) and 5 (User speed range detection) since data processing is different from one use case to another.

3.2.3 Algorithm choice and Model designing

One way to solve more than one task, in a joint manner, is the use of Multi-Task Learning. In ML, we use the concept of Multi-Task Learning (MTL), when a single model is used to solve a series of related tasks (by task we mean: classification, regression, prediction, etc.). The idea of MTL is inspired from human learning activities. As humans, we often use the same basics that provide us with the necessary skills to learn several tasks and master more complex techniques.

The MTL architecture we investigate is based on Deep Learning algorithm (MTL is detailed in chapiter 6). The DL choice is explained by the fact that we are dealing with a lot of data. Once deployed this model should deal with thousands or millions of users at the same time. Actually, according to [31], and [32], Deep Learning's performance continues to improve when more and more data is used for training to outperform the traditional models and algorithms of ML. Furthermore, DL is also appropriate for problems where modeling relationships between large number of features are not tractable. This is the case for the problem of detecting simultaneously the user environment and the user mobility state. Indeed the model has to extract the complexity and variety of various situations met by mobile users. We also opt for a Feedforward Neural Network (FNN) which is more appropriate to detect both attributes (environment and mobility) at same time.

The three key choices to build the model are 1) Last Layer Activation, 2) Loss Function and 3) Optimization Configuration. Table 3.1 can help for choosing a last-layer activation and a loss function for a few common problem types. As for the choice of optimizer (algorithm optimizing the cost function $J(\theta)$ and generally based on gradient descent), for a first try, in most cases, it's safe to go with $rmsprop$ and its default learning rate. The learning rate is a hyperparameter that controls how much to change the model in response to the estimated

error each time the model weights are updated.

Problem type	Last Layer activation	Loss function	Examples
Binary classification	sigmoid	binary crossentropy	Dog vs cat
Multiclass/ single-label classification	softmax	categorical crossentropy	MNIST has 10 classes single label (one prediction is one digit)
Multiclass/ multilabel classification	sigmoid	binary crossentropy	News tags classification, one blog can have multiple tags
Regression to arbitrary values	None	mse	Predict house price(an integer/float point)
Regression to values between 0 and 1	sigmoid	mse or binarycrossentropy	Engine health assessment where 0 is broken, 1 is new

Table 3.1 – Last-layer activation and loss function combinations for training a first model (for a few common problem).

Scaling-up, Optimization and validation

Once the obtained model has statistical power, the question becomes, is it sufficiently powerful? Does it have enough layers and parameters to properly model the problem? Remember that the universal dilemma in machine learning is to find the balance between optimization and generalization. That's to say, the ideal model is one that stands right at the border between underfitting and overfitting, between undercapacity and overcapacity. To figure out where this border lies, first we must build a model that overfits. Then, the next stage is to start regularizing and tuning the model, to get as close as possible to the ideal model that neither underfits nor overfits. In the following we enumerate

the methods that we have used in order to scale up and optimize our models for the separated tasks (user environment detection and user mobility state detection) and also the joint detection of both tasks.

Hyperparameters Tuning

The most delicate and also difficult step (time and resources consuming) in DL application is to select the best hyperparameters set for the model [68]. Hyperparameters per definition are the configurations set to provide to the learning algorithm before a training phase to control the progress of this phase and define the structure of the algorithm. The hyperparameters, that influence the final precision of a given neural network, are mainly: the topology of the network (number of layers, and number of neurons per layer), the size of the batch, the initialization function, backpropagation algorithm, learning step and activation functions (for more details see table 3.2). The major challenges facing the selection of hyperparameters are:

1. The evaluation function of a hyperparameters set combination is very greedy in terms of execution time and allocation of resources, especially in the case of a complex system such as neural networks.
2. The configuration space is too complex and of big dimensionality.
3. In most cases, the regularity of the evaluation function is unknown, which makes this task purely experimental and is based on the nature of the problem and the available dataset.

There are four strategies for tuning hyperparameters [69] that we have tested during our work. A comparison between these approaches is addressed in appendix A for the two tasks of the user environment detection as well as the user speed range detection. These strategies are as follow:

1. **Manual Search:** it is essentially based on human intuition and experience. The specialist manually chooses the values to assign to the hyper-

Hyperparameter	Description	Possible valus
Activation function		Relu, Tanh, Sigmoid, softmax, Swish (see section 2.1)
Layers Number	The number of layers in the network that represents its depth	Integer
Neurons number per layer	The configuration of the neurons number per layer	Integer
Optimizer	Backpropagation algorithm dealing with for weights and biases updating according to the training error.	RMSprop, Adam, Stochastic Gradient Descent, Adagrad, Nadam, etc.
Initialization function	The function that initialize weights and biases during the training phase (the start value).	Normal distribution, Uniform distribution, Lecun function, Glorot fonction, etc.
Epochs	The number times that the learning algorithm will work through the entire training dataset.	Integer
Batch	The number of samples to work through before updating the internal model parameters.	Integer that depends on the total number of samples in the dataset

Table 3.2 – Non exhaustive hyperparameter list for neuronal networks with their possible values examples.

parameters. Then, with each evaluation they adjust values, towards the direction that optimizes the most the model's accuracy. They repeat this process until you they run out of patience or they are satisfied with the results.

2. **Grid Search:** It is a basic approach for finding hyperparameters. It consists of going through all the possible combinations of these variables,

and evaluating the accuracy of the model for each set of values and selecting the one that gives the best result. The major drawback of this method is the huge execution time it takes to exhaustively evaluate all of the possibilities, but on the other hand, convergence to the best set of values is guaranteed.

3. **Random Search:** Random search has been set to overcome the drawbacks of the grid search (greediness) [70]. It aims to sample uniformly from the configuration space without making exhaustive evaluations, which lightens the search load. It sets up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources.
4. **Bayesian Optimization** [71]: Unlike other hyperparameter optimization methods, Bayesian optimization uses knowledge of previous iterations of the algorithm. With Grid Search or Random Search, each set of values is chosen independently. But with the Bayesian method, each time you select and try different hyperparameters, you get closer to perfection.

In most of our experiments and presented results in this thesis, Bayesian optimization has been considered for tuning the set of hyperparameters. Based on a probabilistic model of the objective function called the surrogate function, the Bayesian optimization reduces the frequency of calls to the the objective function, to the lowest possible. The surrogate model is represented by the probability of the score knowing a fixed hyperparameters' set ($P(score \mid hyperparameters)$). Actually, the use of a surrogate model enables the algorithm to select the most promising hyperparameters for the objective function evaluation, so that, the search does not spend a significant amount of time on looping on bad combination of hyperparameters. That is to say, such optimization leads to a faster convergence of the Deep Learning based model.

By means of the surrogate model, which is easier to optimize, the algorithm

selects the most promising hyperparameters' set to be evaluated in the objective function so that the search does not spend a significant time on trying bad sets. The surrogate model is represented by the probability of the score knowing a fixed hyperparameters' set ($P(score \mid hyperparameters)$).

The Bayesian optimization algorithm is structured as follows:

1. Build the surrogate model of the objective function.
2. Selecting hyperparameters that give good scores on surrogate model.
3. Calling the objective function on selected hyperparameters.
4. Update the surrogate model including the new results.
5. iterating over 2-4 until the maximum number of evaluation is reached.

There are multiple formalizations of the Bayesian optimization that differ in their way of building the surrogate model and in the selection criteria of the next hyperparameters to evaluate. The most used option for the selecting criteria is *"Expected Improvement"*(EI). EI is given by the following formula:

$$\begin{aligned} EI_{y^*}(x) &= \int_{-\infty}^{+\infty} \max(y^* - y, 0) P(y \mid x) dy \\ &= \int_{-\infty}^{y^*} (y^* - y) P(y \mid x) dy \end{aligned}$$

with :

— y^* is the threshold value of the objective function.

— x is the proposed set of hyperparameters.

— $P(y \mid x)$ is the surrogate probability model.

Several choices are presented for building the surrogate model like Gaussian processes, random forest regressions and tree parzen estimators. Tree parzen estimator (TPE) is the most used algorithm for modelling the objective function over ML community, it consists of constructing $P(y \mid x)$ from $P(x \mid y)$ given the

Bayes theorem that links them together:

$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})}$$

$P(\boldsymbol{x} \mid \boldsymbol{y})$ is computed over the history of pairs of the set of hyperparameters and the objective function score.

$$P(x \mid y) = \begin{cases} l(\boldsymbol{x}) & \text{if } y < \boldsymbol{y}^* \\ g(\boldsymbol{x}) & \text{if } y \geq \boldsymbol{y}^* \end{cases}$$

For every iteration, TPE proposes a new $\boldsymbol{x}$ (a set of hyperparameters) by drawing from $l(x)$ (the distribution that yields lower scores than the threshold of loss y^*), save it in the history as a pair (x , y) where y is the score, and then using that up-to-date history it builds the new $l(x)$ and $g(x)$ to come up with the probability model of the objective function (for further reading see [69]).

Regularization methods

Dropout Layers Adding dropout [72] layers is one of the newest and most effective regularization techniques for neural networks. This technique, invented by Google, is inspired by the bagging method [2]. Actually, training and evaluating multiple neural networks, as proposed in the bagging method, are an enormous burden in terms of execution time and memory used for such complex models. The Dropout provides an approximation less greedy than training and evaluating an exponential set of neural networks (the case of bagging method). In other words, dropout is a regularization method that approximates training

2. Bagging is a regularization technique which consists in reducing the effect of overfitting by taking several predictors of the same type, training them and retrieving the prediction by making the weighted sum of their outputs or by majority vote.

a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or dropped out. This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In fact, each update to a layer during training is performed with a different view of the configured layer. The figure 3.4 better explain the dropout mechanism.

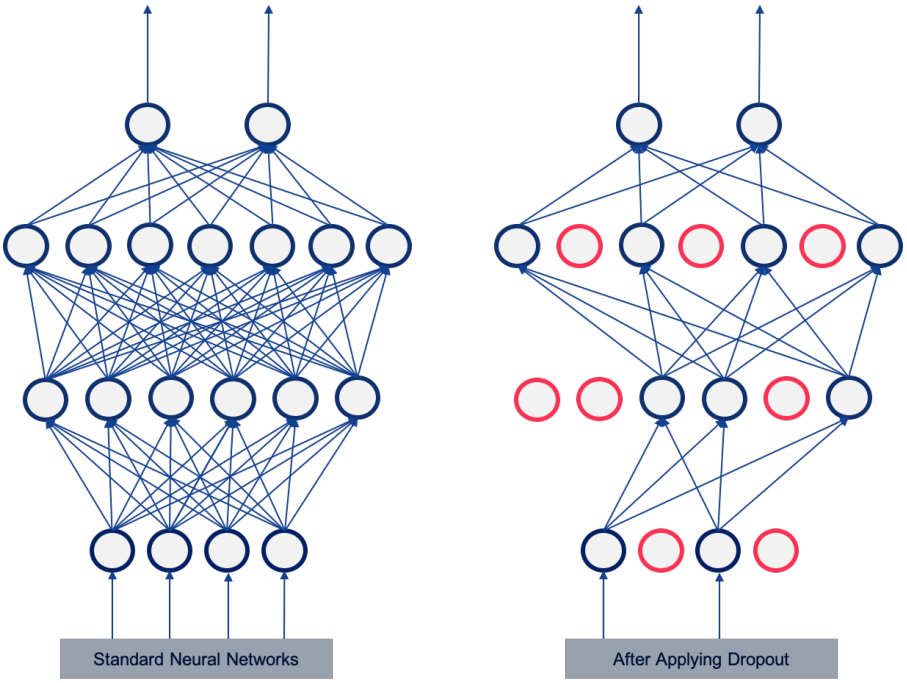

Figure 3.4 – Dropout mechanism in neural networks

Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. As a result, models trained with dropout layers have been very successful in making these models more robust at low cost. Dropout roughly doubles the number of iterations required to converge. However, training time for each epoch is less.

Adding Noise It is a classic and a very basic regularization method. In [73], the authors have shown that by allowing a little random margin of inaccuracy in Feed Forward Neural Networks, a model can perform better on both training and accuracy. Neural Networks are able to model functions that change their outputs spectacularly in response to just a small input variation. Some noise is introduced to the neural network model via a Gaussian Noise layer during the training phase. Noise permits to have some robustness in the output and smoothed decision borders. Thus, noise helps reduce the chance of over-fitting, even when training dataset is small, and aids in the generalization of the model. Thus, we add a Gaussian Noise layer inside our model. It consists in adding noise to output of the layer before the activation function. The added random value, ϵ_i, follows a normal distribution $\epsilon_i \sim \mathcal{N}(0, \Sigma)$, i.e., with a mean equal to zero and the co-variance matrix of features Σ.

Early Stopping While training a large network, there will be a point during training when the model will stop generalizing and start learning the statistical noise in the training dataset. Thus, the challenge is to train the network long enough that it is capable of learning the mapping from inputs to outputs, but not training the model so long that it overfits the training data. Therefore, we have to force stopping the training if this point is reached, but not the epoch's number (see figure 3.5). Thus, to obtain the most robust model with the best accuracy in training as well as in test, it is necessary to record the parameters learned progressively (in every epoch). With every recording, it's necessary also to check that the model is not to going far in the generalization and to check also that the model is getting better with every step. Otherwise, if it's not the case, it's necessary to stop the learning since no parameter is improving the best

validation error recorded for a certain number of predetermined iterations in this region. This method called EarlyStopping, is the most intuitive and popular method in regularization.

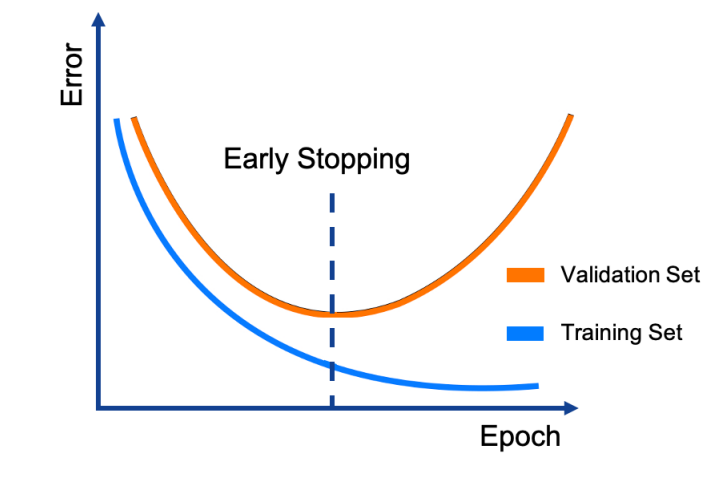

Figure 3.5 – The training error and validation error curves during the learning phase in order to visualize the Early stopping point.

Validation

Once we have built the best optimized model we evaluate it one last time on the test set. If it turns out that performance on the test set is significantly worse than the performance measured on the validation data, this may mean either that the scaling up and optimization procedure were not reliable, or that you began overfitting to the validation data while tuning the parameters of the model. In this case, we have to investigate more the validation data set (either it is not enough, or it is poorly sampled compared to the training set, etc.). Otherwise, we have to switch to a more reliable evaluation protocol (such as iterated K-fold validation or adding regularization layers, etc.). If it is still not working we return to the data step (It's the cornerstone of the ML).

3.3 Generation of high quality and representative data

Generally, to use any ML approach, we have to guarantee complexity and data. Complexity shall be accepted mainly for the training phase. Data is the most important fact : it shall be representative and good quality. So it does not matter if you don't know which algorithm or model of machine learning you want to use because if you have data then you can choose your algorithm or your architecture after some preliminary statistics conducted on this data. Besides, the more data we have, the better model we will have and for more data we have to accept complexity and heavy computing. Thus, to sum up, we can say that data is the cornerstone of any machine learning study because without data we can't even think about machine learning solutions.

3.3.1 Data collection modes

Currently, there are many modes to collect or gather data. The choice of collection mode depends on the research themes and objectives (target population, sensitive subject and/or sensitive population), methodological and logistical constraints, budget and time constraints. In telecommunications field, we can mainly enumerate 3 ways which are widely used to collect data : (i) simulation, (ii) log reporting, (iii) drive tests and an emerging method largely used by application providers : (iv) crowd-sourcing :

- **Simulation:** Using simulation, data is generated via a mathematical or a stochastic model. With simulation, it is very easy to get abundant amount of data. Such data can be used to quickly develop or test new ideas or new algorithms and tools, particularly for never-before-tried applications or use cases. Instead the real collection of data, simulation tools are easily customized to allow the collection in the case of new tasks/ envi-

ronments/ features/ etc. However, it may take incredibly long time and it will be incredibly expensive to collect data in real settings each time to be adapted to a minimal change. On the other hand, since ML aims mainly to find a mathematical relationship between input and the output thus using simulated data based on some mathematical models may be prone to showing bias towards these models. Thus, it is not the best choice to train a machine learning algorithm that will be deployed in production settings. Yet since the data is not real, it differs in key ways from realistic data, bringing many risks associated with convergence, fine-tuning algorithms, feature engineering, etc.

- **Reporting and logs:** This mode is widely used in many fields for analytics like e-commerce, phone applications, servers, cloud infrastructures, IoT, mobile devices, etc. Logs and reports analytics involve searching, analyzing, and visualizing all the network, all the IT systems and the infrastructure to gain operational insights. Generally, this mode of data collection is used in understanding what has happened in a system (bugs, problems, etc) and derive useful metrics in monitoring and performance handling[74]. For example, [75] proposed a ML approach to prevent malicious calls over telephony networks based on a large-scale call log database. In [76] network layer traces and logs to help labeling their dataset are used in order to classify flows and buffer state for YouTube's HTTP adaptive streaming service in mobile networks. [77] used a big dataset coming from iphone logs to evaluate an anomaly-based intrusion detection systems for mobile devices using machine learning classifiers.
- **Drive tests:** This mode is widely used to collect data in telecommunication field. It is a mode in which operators often send engineers in the terrain with a vehicle or some people equipped with sensors moving in

specific places to collect radio measurements, to discover problems such as coverage holes in the network, user location and to determine whether certain parameter tuning is needed [78]. Traditional drive tests systematically obtain measurements using vehicular infrastructure such as roads which does not necessarily account for the entire situations of user experience. Conventional drive test is a manual process. To collect network quality information, an operator often needs to send engineers directly to the concerning area and obtain radio measurements in a hand-operated manner. With such measurement vehicle, engineers would perform test calls in the car and record measurement results along the drive route. For indoor coverage engineers perform "drive tests" on foot, using specially developed measurement equipments that can be carried by hand. Drive test enhancements have been introduced in the 3rd Generation Partnership Project (3GPP) standards for LTE [79]. The concept of Minimization of Drive Tests (MDT) was introduced in Release 10 which automates the data gathering enabling the use of network related User Equipment (UE) data. Moreover, this approach provides measurements of where users are geographically located during attachment to the base station/network. However, this controlled mode imposes limits on capturing the reality. Such data collection campaigns are run for limited hours per day, during short periods (couple of weeks) and at some specific places. For user behavior prediction, using the data collected in drive test mode will not deliver a general model, since, we will miss big sides of users' behavior, for example the case of his night life, etc. A comparison between this mode and the crowd-sourcing mode performances using ML approach will be detailed in chapter 4.

- **Crowd-sourcing:** In the last years, crowd-sourcing approach gained momentum as a viable strategy for solving very large-scale problems with

the help of the masses [80], [33], [81]. It is an emerging measurement method which relies on user participation in the measurement process. The task of collection is outsourced to the crowd, thus cost-effective network monitoring can be performed at a societal scale, using a possibly large number of end users' devices scattered over a wide geographic area. In crowd-sourcing the solution is usually provided by distributed and decentralized agents running on end-user devices and performing active, passive or both types of measurements in real time. Largely adopted big firms like Google and Netflix, the crowd-sourcing mode allows to gather data more cheaply and in large quantities. This data are stored and cached in special platforms dedicated to data reporting controlled and managed by a tenant (e.g. service provider or application provider). Scalable cost effective approach that captures real end-user experience crowd-sourcing mode collects measurement data from the end-user side also allows usage profiling for user behavior monitoring. It enables to better reflect the users' behaviors by capturing a huge diversity of their experiences. A drawback of this method is that the measurement agents are restricted by the OS APIs. Also, incentivizing the users for contribution can prove challenging as users should agree and should themselves install the necessary software. Instead drive-test collection mode that imposes measurements in limited situations generates data with a poor representation of a daily life of a mobile user.

3.3.2 Data collection description in our study case

For this work the used dataset has been collected in a crowd-sourcing mode. Indeed, due to their small size and popularity, the portable phone devices are almost always with users 24h/24h and 7d/7d during their various activities while moving or not. The crowd-sourcing mode enables us to build a dataset for

training and evaluation as close as possible to the complexity and the variety of usage situations, of mobile users and their movement in real world.

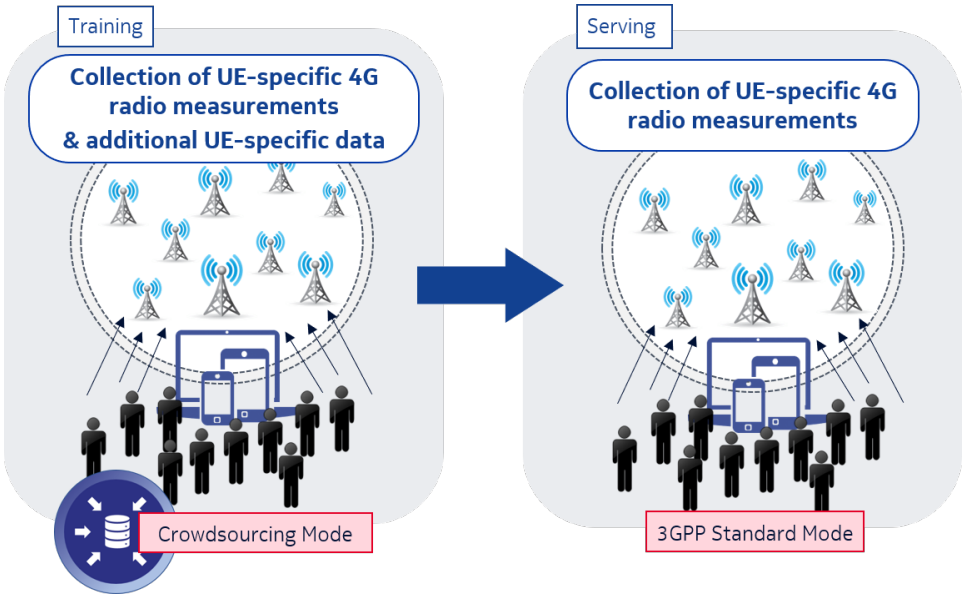

Figure 3.6 – Data collection scheme for training and serving phases

As shown in Fig. 3.7 and described in [40], our dataset has been collected:

— at different locations (many cities and places) in France (red dots) and of course on the roads linking these locations where users are moving with different speeds
— in various location types (mountain, beach, forest, cafès, streets, mall ...)
— in all weather types (heavy rain, foggy, sunny, snowy, windy ...)
— in diverse cities/places (villages, metropolis ...)
— in various cell deployments (macro, micro, small ...)

The data has been collected during almost 2 years, 24h/7, with an average of 1 measurement per 15 seconds, when the mobile phone session is active, and 1 measurement per 2 minutes otherwise.

The recording of data has been done in the mobile terminal using a mobile application software. For the training phase, the crowd-sourcing mode also includes in addition to UE-specific 4G radio measurements UE-specific metadata

that is required to do the labelling in supervised or semi-supervised mode (Fig. 3.6). Indeed, this metadata permits to label the output used to train the model in these approaches. For the serving phase (using the model to infer the user environment or the mobility state) only UE-specific 4G radio measurements is collected via 3GPP standard process.

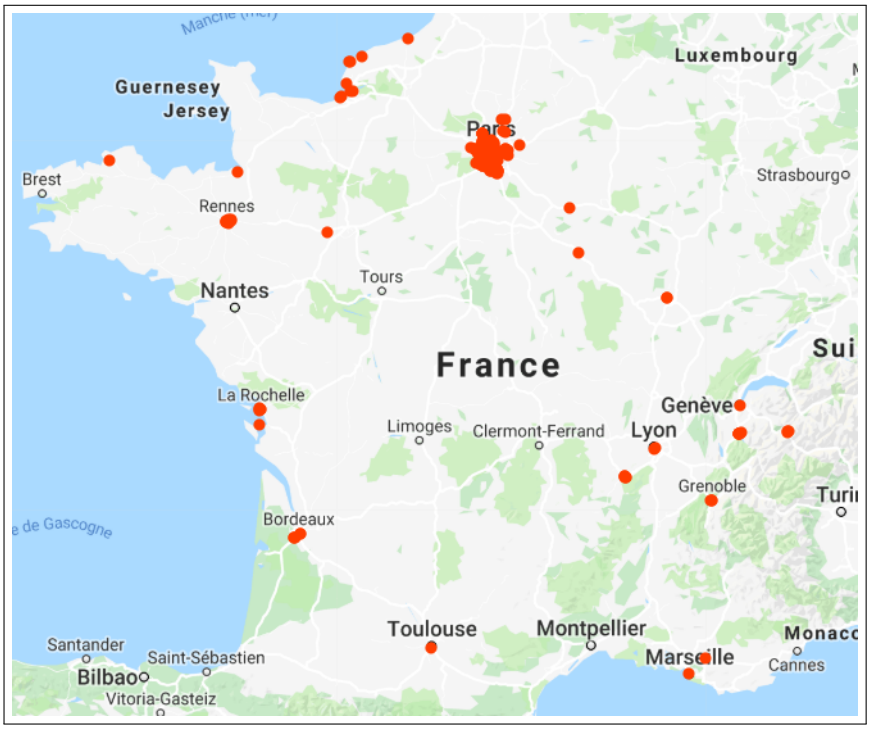

Figure 3.7 – Data collection points in France: multiple places

3.4 Conclusion

In this chapter, we have detailed the general workflow of any machine learning process in general and any Deep Learning process in particular. This workflow is composed of 5 main steps 1) Identifying the problem and analyzing the needs 2) Generating good data 3) Analyzing and cleaning data 4) Identifying the appropriate algorithm and training 5) Scaling up, optimization and Validation. We also defined some retroback action in case that for a given step we don't get

the expected performance. In the rest of the chapter, we tried to project our use case into this workflow. In this projection we have fixed some basic points to follow during solving both tasks dealing separately with the user environment detection and the user speed detection and also the joint detection of these two user behaviors. Finally, we have presented our dataset obtained from a crowd-sourcing collection campaign. This dataset will be our cornerstone to resolve, separately or simultaneously, both tasks of user environment detection and mobility speed range detection, while consuming a service.

	Type	Explanation
Hybrid Learning	Semi-supervised Learning	Used when the training data contains very few labeled examples and a large number of unlabeled examples so we can make effective use of all of the available data, not just the labelled data like in supervised learning [1]. This Learning Type will be detailed in 4 in section 4.4.
	Self-Supervised Learning	Called also self-supervision or pretext task. It is a relatively recent learning technique (in ML) where the training data is autonomously labelled e.g. can be labelled by finding and exploiting the relations (or correlations) between different input signals (coming from different sensor modalities) [82].
	Multi-Instance Learning	It is a form of weakly supervised learning where training instances are arranged in sets, called bags, and a label is provided for the entire bag. It is gaining interest as it naturally fits various problems, allowing to leverage weakly labeled data [83].
Others	Multi-Task Learning	In machine learning, we use the concept of Multi-Task Learning (MTL), when a single model is used to solve a series of related tasks (such as: regression, prediction, etc.). It mainly aims to leverage useful information contained in multiple related tasks to help improve the generalization performance of all the tasks [84]. This Learning Type will be detailed in 6 in section 6.1.
	Active Learning	Active Learning is a special case of Supervised Machine Learning. This approach is used to construct a high performance classifier while keeping the size of the training dataset to a minimum by actively selecting the valuable data points. The main hypothesis in active learning is that if a learning algorithm can choose the data it wants to learn from, it can perform better than traditional methods with substantially less data for training [85].
	Online Learning	Online learning involves using the data available and updating the model directly before a prediction is required or after the last observation was made. Online learning is appropriate for those problems where observations are provided over time and where the probability distribution of observations is expected to also change over time. Therefore, the model is expected to change just as frequently in order to capture and harness those changes [86].
	Transfer Learning	Transfer learning is a type of learning where a model is first trained on one task, then some or all of the model is used as the starting point for a related task. It is a useful approach on problems where there is a task related to the main task of interest and the related task has a large amount of data [32].

Table 3.3 – Different Types of Learning in Machine Learning derived from the 3 basics ones (Supervised, unsupervised, Reinforcement Learning)

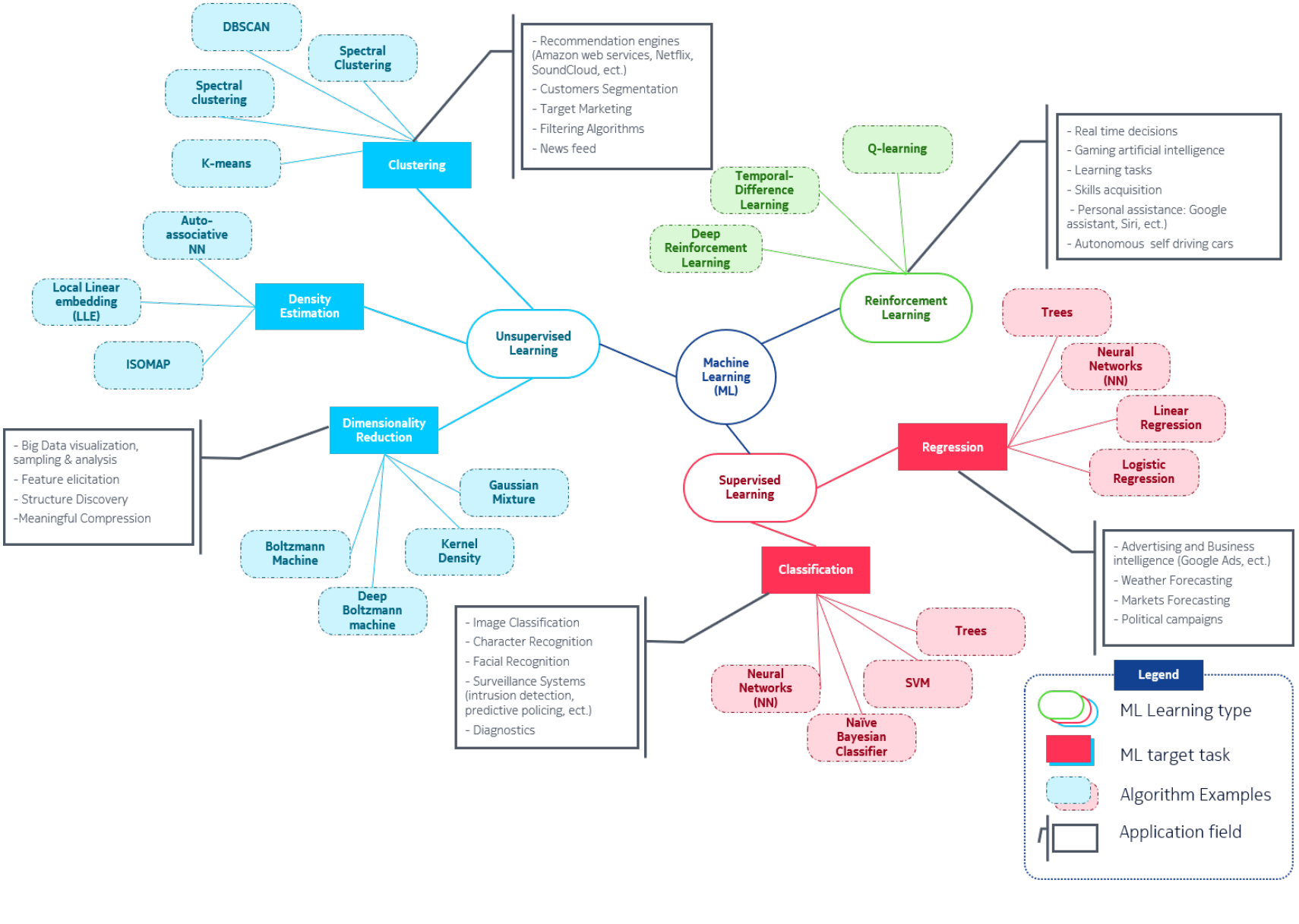

Figure 3.8 – Machine learning categorization [87]: Different machine learning types and applications in computer science fields.

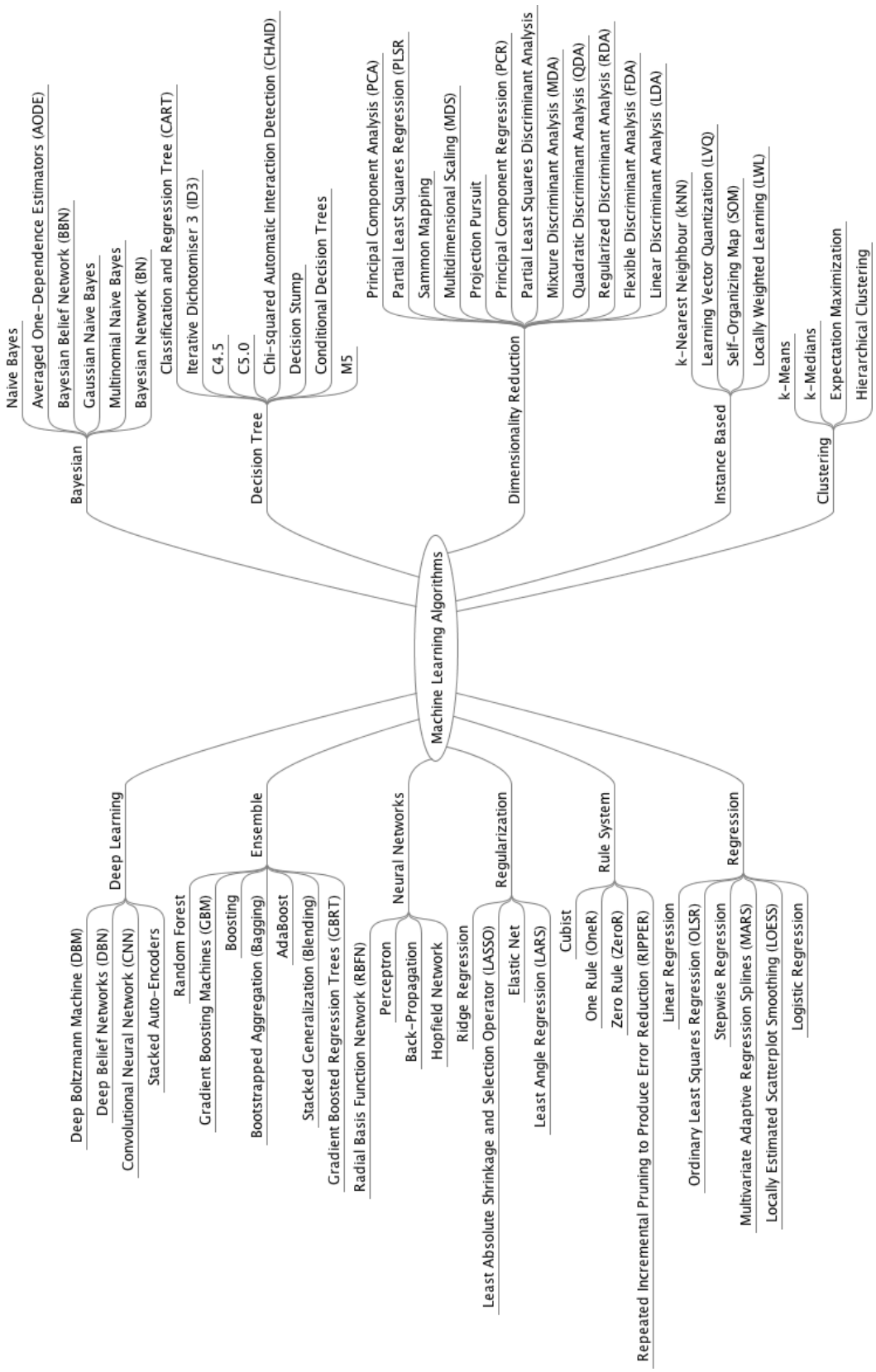

Figure 3.9 – Machine Learning Algorithms Mind-Map [50]

Chapter 4

User Environment Detection: Where is the mobile user while experiencing a service?

As a first step towards the unified AI-based model for a joint detection of all attributes of a user behavior, we tackle the detection of first behavior attribute in this chapter. This attribute deals with the user environment. In this study, we aim to answer the question "Where is the mobile user while experiencing a service?". The investigation of this behavior is studied progressively going from detecting just 2 types of environments, i.e., Indoor Outdoor Detection (IOD), to multiple environments detection (eight environments). We call this multiple environment detection as user environment detection (UED). On the other hand, IOD refers to the estimation of 2 types of mobile users' environments, that is to infer whether a user is Indoor or Outdoor. UED is a more general case of IOD, where we aim to investigate the environment with more granularity (with more than 2 classes: Indoor and Outdoor). IOD, and more generally UED, is a cornerstone of the user behavior contextualization, which in turn can be used for adapting mobile network resources, or improve the user location etc [20], [88]. The idea is to have more information on the user by knowing his or her environment type. The situations where mobile users request service are more complex than being described by two types of environment. The two classifica-

tion states (Indoor or Outdoor) don't reveal the complexity of the daily activity of mobile users. Let's take an indoor environment for example. The user behavior can be affected by the type of indoor environment. As a matter of fact, a user at work does not consume the same type of services as another user at home or a user in a cafe. These categories *work*, *home*, as well as *café*, are indoor environments, but the user behavior, in terms of consumption of mobile services, differs from one environment to another. Hence, there is a need to investigate the user environment in more detail and granularity. Besides knowing more precisely the environment type can help to enhance the prediction of the type of the preferred application consumed and enable enhanced QoS in resource management and optimization.

Our hypothesis is that IOD or UED can be performed accurately, automatically and in real time using machine learning techniques. However, this in turn needs data for learning. As explained in chapter 3, data collection is the first step of designing IOD or more generally an UED solution based on machine learning. In this chapter, we compare the impact of data collection, using drive test mode vs using the crowd-sourcing mode, on the ML performances. The crowd-sourcing-sourcing approach is becoming popular for collecting and analyzing real and large network measurement datasets coming from mobile phones or any other connected devices. Data obtained from smartphones also contains the natural mobility vector of people carrying them. This ensures cost-effective, continual and spatio-temporal monitoring.

To the best to our knowledge, the user environment detection (UED) issue has not been studied extensively in the literature. So far, what has been carefully studied in literature, is UED as a binary classification or detection problem, with only indoor and outdoor as classes (what we are referring to as IOD). In the state of the art, most of IOD works mainly use the received signal power $RSRP$ and the received signal quality $RSRQ$ as input for the IOD model. Actu-

ally, both of these signals are highly correlated to user environments. However, using only them for IOD is not sufficient to guarantee IOD's good performance. Alone, they are not enough to reach our target success criteria, especially while facing ambiguous measurement points in mobile environments or in ambiguous user locations relative to eNB. Hence, there is a need to vertically expand the dataset used to solve the IOD issue by adding other signals. In other words, one should look for additional signals to solve the IOD issue and not rely solely on signal power or quality. Therefore, we propose to use new input signals which are related to the radio signal quality, the user location and mobility. For this purpose, we use the following signals: IOD uses Reference Signal Received Power ($RSRP$), Channel Quality Indicator (CQI), Timing Advance (TA) and Cell Id (Cell Identifier). They represent 3GPP defined signals or indicators. $RSRP$ and CQI are measured by the phone device and sent to eNB via standardized protocols. TA and Cell Id are derived inside the network when a user is connected.

Note that IOD or UED, performed in the network, should perform automatically and consider the constraint of minimal human intervention. For this, we propose to study supervised and also semi-supervised Deep Learning-based classification methods for training automatically IOD or UED classifiers. Both methods require labeled data for the training phase. However, the labeled data, used for ML training, is either tagged manually or automatically. Manual data tagging can be expensive and complex and even unfeasible for certain mobile operators, if they have to tag all the collected data. But, semi-supervised methods are a mix of supervised and unsupervised approaches which can learn from partially labeled dataset. Such method reduces human intervention to the minimum possible. We investigate therefore three semi-supervised methods that 1) learn from both labeled and unlabeled data and 2) make use of information on received signal power, signal quality, distance and mobility. The promise of

semi-supervised learning is that we can get our ML algorithm to learn from "unlabeled" data, which in turn is easier to obtain. An unlabeled example is mostly less informative than a labeled example. Nevertheless, we can get tons of such less informative examples, by collecting huge crowd-sourced unlabeled signals. Now, if our algorithms can exploit most of the unlabeled data effectively, then it will enable us to learn more possible environment types related to the user behavior. That way the data will closely reflect the users' habits. This reminds us that today, in the domain of ML, Deep Learning surpasses all classical ML algorithms [31],[32]. It is said in [89]:"The more data we have, the wider the Neural Network is, the better the performances are."

This Chapter is organized as follows: **Sec. 4.1** describes the main IOD and UED works in literature. **Sec. 4.2** details the inputs or features used for IOD or UED. **Sec. 4.3** provides first a preliminary data analytics, then compares the two data collection modes crowd-sourcing and drive-test and finally shows the impact of replacing $RSRQ$ by CQI and adding new features of mobility or distance indicators on IOD performances with supervised ML. **Sec. 4.4** presents results of IOD with semi-supervised learning approaches. **Sec. 4.5** deals with environment detection considering more than 2 classes (UED). In **Sec. 4.6** we investigate the impact of labeled data and the unlabeled data volume on the performances of the semisupervised approach for the user environment detection. Finally conclusion is presented in **Sec. 4.7**.

4.1 State of the Art

As mentioned before, the user environment detection (UED) issue has not been largely studied. So far, what has been carefully studied in literature, is IOD. Proposed solutions for IOD are usually divided into two categories [90]. IOD is either considered as a statistical issue where a weighted score or a

threshold is defined to determine the mobile environment, or as a classification problem sorting mobile users between multiple classes. In most of these works, only two classes are considered (Indoor/Outdoor), but in some works, three classes are selected (e.g. Indoor/Semi-Outdoor/Outdoor). The Fig. 4.1 illustrates the classes used.

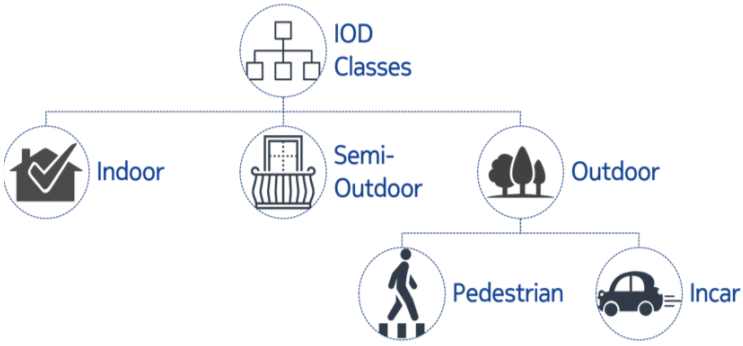

Figure 4.1 – IOD classification scheme: in 3 main classes, according to the state of the art.

In addition to such categorization, the IOD problem can also be distinguished based on the location where IOD is performed, either at the mobile terminal side (that's to say using phone sensors data) or at the mobile network side (that's to say using only the data known by the mobile network). In the following, we highlight some of the works dealing with the IOD issue, presenting them according to this classification: whether data is known by the network or not.

In first category, which is based on signals coming from the mobile phone sensors, [91] authors propose to use a threshold of a signals set collected from some phone sensors related to: radio signals, cell signal strength, light intensity as well as the magnetic sensor to infer whether the mobile user is indoor or outdoor. Like [91], [88] also addressed IOD using the same set of sensors combined with some more like the sound intensity, battery temperature and the

proximity sensor. The investigated IOD solution is based on ML algorithms and more precisely a semi-supervised ML approach. Their solution, implemented on different android devices shows a 92.33% of accuracy and provides the highest detection performance in comparison with existing methods including supervised classifier. This solution shows the interest of using semi-supervised ML approaches for IOD. Thus, this motivates us to try similar solutions on the network side.

In the second category, which is based on signals known at the network side, in [21], authors have considered IOD at network side as a classification issue. Once the indoor or outdoor location of a user is detected, it is combined with other signals to localize the mobile user by estimating its longitude and latitude in a more accurate way. For the IOD classification task, they used $RSRP$ and $RSRQ$ signals and tested many algorithms: Support Vector Machine (SVM), logistic regression and random forest. SVM was the solution retained since it performed best. However using only $RSRP$ and $RSRQ$ signals for IOD is insufficient since these two signals are correlated to each others. Besides $RSRQ$ is not frequently sent to the eNB which can cause problems in real-time inference of IOD. That is why, in our works, we propose to replace the $RSRQ$ which is a quality signal by another quality signal CQI.

In [92], the authors optimize the use of radio measurements in wireless networks. They use radio signal measurements collected in different situations of mobility, with varying speed (low, medium, high). They dynamically estimate the signal attenuation. This in turn helps them to efficiently classify the mobile user environment (pedestrian, in-car or non-moving) and it finally improves the handover process. This confirms that a user's mobility is strongly correlated to his or her environment. The authors assume that once the signal power attenuation is estimated correctly, we can easily classify whether the mobile user is pedestrian, in-car or unmoving. This is because the measured power

signal for an unmoving user does not show too many variations unlike the in-car or pedestrian cases. Nevertheless, this proposition is still at an early stage and it has not been thoroughly developed yet.

As for works in [20], they use a combination of phone sensors as well as the signals coming from the network. Authors use a Bayesian detector that combines the signals measured from the cellular ($RSRP$ provided by the cellular modem) and $GNSS$ receiver (the confidence radius of the location, provided by the active localization sensor). This information provided by both measurements is combined with a joint posteriori probability based on the distributions of $RSRP$ and GPS measurements to perform Indoor/Outdoor detection.

Among the signals, which we are using in our work, we propose to add another feature that is correlated with the user mobility. Many works in literature have addressed the estimation of user mobility. In [93], authors use $RSRP$ measurements to capture the signal's speed dependent and shadowing induced time variations in order to compute the UE speed. They propose two methods: one based on a spectral analysis and another based on a time-based spectrum spreading. For both methods, the variation is compared to a reference curve or a look-up table (database) and the difference is analysed to compute the UE speed. In [94], authors propose a method for estimating the UE mobility, that relies on UE's history information about the UE cell sojourn time. The neighbouring eNBs exchange among them the learned network topology as well as the UE sojourn time history. Using such information, the eNBs classify the speed to one of the three mobility classes defined by 3GPP. Both methods for mobility estimation in [93] and [94] have shown good results, however they estimate the speed of UEs only in some specific use cases. In addition, they are complex to setup.

For us, the issue in this chapter is not to study the mobility itself (a detailed state of the art dealing with the mobility is addressed in chapter 5), but

rather to exploit a simplified mobility indicator in order to improve the IOD performance. For this reason, we employ the standardized 3GPP procedure of mobility estimation. Actually, this low complexity approach is advantageously simple. But, according to literature, the 3GPP procedure is not precise enough for an accurate mobile speed estimation [94]. However, in our study, we aim to evaluate whether the mobility indicator, as an additional input, can bring enough rich information to improve the performance of IOD system.

In [95], [96], the user mobility is estimated and classified to one of the three categories (Normal, Medium, High). This estimation using standard 3GPP procedures is done as follows:

— 1- Compute the number of handovers or cell re-selections (denoted by NCR) during a sliding time window (denoted by TCR_{max}).

— 2- If a UE's NCR count is smaller than a threshold NCR_{medium}, then the UE's mobility state is determined as "Normal". If the UE's NCR is greater than NCR_{medium}, but less than NCR_{high}, the state is determined as "Medium". Finally, if the UE's NCR is greater than NCR_{high}, then the state is determined as "High".

4.2 Data features

In this section, we detail the data (signals) that we use to solve the IOD and UED issue.

In machine learning domain, data collection is the first main step for building the desired knowledge about the user environment. For this goal, we opt for a dataset composed of radio signals ($RSRP$, CQI), temporal features (TA, Time), a Mobility Indicator (MI), and finally the environment label. Thus, our data-set is composed of a vector of following 6 features:

— Time: recording time of signal or burst data arrival (ms), the time of mea-

surement will help us to optimize the ML training mainly in the case of semi-supervised Learning.

— $RSRP$: the average received power of the Reference Signal (RS). The $RSRP$ value lies between -140 dBm to -44 dBm [97].

— $RSRQ$: the ratio between $RSRP$ and $RSSI$ (Received Signal Strength Indicator) which lies between -19.5dB and -3dB [97], that represents the total power of the received signal (including the transmitted signal, the noise and the interference). $RSRQ$ is not a feature used in our AI model, but we will use it for fair comparison between our proposal and the state of the art. Instead of $RSRQ$, we use another quality signal CQI, which is frequently sent to the eNB.

— CQI: Channel Quality Indicator is used to indicate the most appropriate transmission modulation and coding scheme to be used [98].

— TA: Timing Advance is used to control UL signal transmission timing [99].

— MI: The number of the Cell ID changes ($NCID$) in a sliding window of a given duration (TCR_{max}) [95] [96].

— Label: Indoor or Outdoor label in case it exists.

To estimate the value of mobility indicator, we use a sliding window of duration TCR_{max}. In urban environments, considering macro-cells, we assume a typical separation distance of around $900m$ between two base stations. Assuming this distance, we estimate that a mobile user with a typical average speed of $30km/h$ will move to another cell, at least once in $100s$, excepting a few rare cases. Consequently, $TCR_{max} = 100s$ of history on visited cells (UE History Information) is sufficient for starting to observe cell ID changes.

However, this is valid only for urban environments. Figure 4.2 plots the time to cross a cell vs. typical user speed for three environment types - urban macro-cell, suburban macro-cell and small cell - assuming a trajectory model

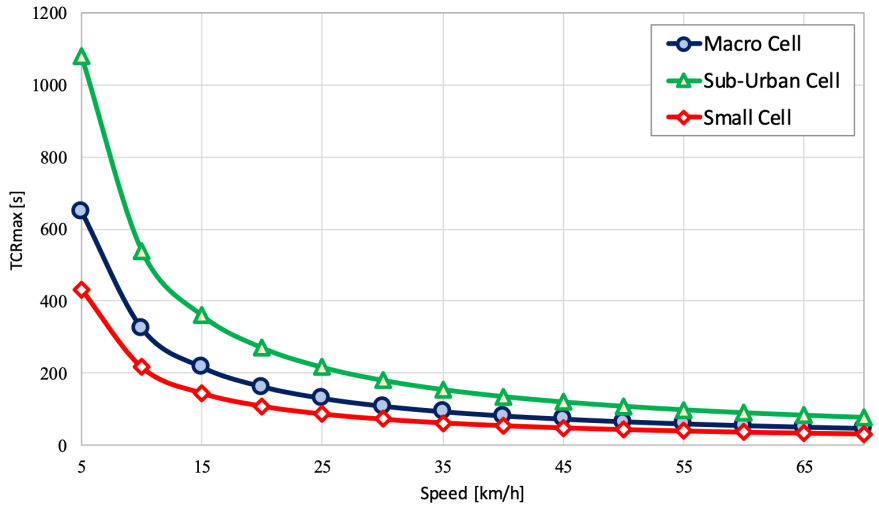

Figure 4.2 – Time to cross a cell (s) vs. user speed (km/h) - urban and suburban Macro-cell and small cell environment cases.

where a user follows a straight line. We assume a distance of 1.5 km as a typical separation distance between two base stations for suburban environments and 350 m for small cell deployments. As expected, we observe that the crossing time is a function of the environment. For a speed of $30km/h$, TCR_{max} value is around 100s. TCR_{max} has a lower value in the case of small cell deployments because such cell-types have smaller radius.

4.3 Supervised Learning-based classification: Indoor Outdoor Detection (IOD)

So far, works on Indoor-Outdoor Detection (IOD) which use machine learning have used two signals to detect the user environment: a signal power measurement $RSRP$ and a signal quality measurement $RSRQ$.

In this section, we evaluate the impact of replacing the used quality indicator in literature $RSRQ$ by another popular quality indicator CQI. We motivate this choice by the fact that CQI is regularly measured by the device and natively

reported to the mobile network, whereas $RSRQ$ reporting needs to be triggered by a specific 3GPP procedure. Thus, using CQI doesn't need to implement additional signaling. We also investigate the impact of adding two additional input features referred to as MI and TA for the IOD task. MI and TA represent respectively the mobility type and the distance between user and the base station (eNB). We expect that these additional metrics can efficiently help to overcome ambiguous classification points.

For the performance evaluation, as explained in chapter 3, we use the accuracy metric which is the ratio of correctly classified instances divided by the total number of instances. We also use the metric F1-score that is by definition the weighted average of Precision and Recall. At the time of experiment, we had collected around 2M lines of data per user. This number is still growing on as of today. In this part dealing with IOD, we have used 250K lines of data which is specific to LTE networks. This dataset is made of around 30% of labeled data (exactly 72k of data volume) and around 70% of unlabeled data (178k of data volume).

We consider IOD as a supervised learning problem. That is to say, the dataset has been restricted to using only the labeled part: only 72k is used. For our experiments, we divided our dataset as follows: 70% for training, 30% for validation and test. In other words, the model is trained and evaluated on data which is split into two subsets composed of 70% of data for training and 30% of data for tests and validation.

To allow for fair evaluation, we compare the performance with the classical ML algorithms. Thus, we compare the performances of the following: Support Vector Machine (SVM) used in [21], Logistical Regression, RandomForest, a clustering algorithm (k-means) and Neural Network/ a supervised Deep Learning. The Deep Learning architecture and hyperparameters were set according to a hyperparameter optimization process that was detailed in Chapter 3. These

results were presented in our papers [39] and [40].

4.3.1 Preliminary study of CQI, mobility and distance impact

$RSRP$ and CQI are radio metrics directly linked to the mobile user environment as they represent the extent of environment attenuation. But, just using them is not enough to correctly classify some measurement points that are referred as ambiguous points. These points are relatively difficult to classify. The ambiguous points belong to multiple situations of user daily activity. Such situations are due to the nature of real life that a user is living in his daily life. For example, consider a mobile user travelling in train. The user is considered as outdoor meanwhile the received signal strength is bad because of not only the high speed of the train (Doppler effect), but also because of the surrounding structures. Indeed, the metal windows and carriages of train cause a significant attenuation of radio signal power. For example, in 2 GHz frequency band, the penetration losses from train carriages are usually in the range of 20 to 35 dB [100]. In crowd-sourcing data, we have many ambiguous situations that need to be handled by ML. As we will see in next sections, the performance of tested ML algorithms does not meet our satisfactory criteria, when not considering additional data features as compared to SOA.

Figure 4.3 shows the indoor and outdoor Cumulative Distribution Functions (CDFs) derived from the crowd-sourced data. The blue curve with full-line represents the data taken in normal speed: corresponding to only from static locations (various types of building crossed) and low speed points. The blue dotted curve depicts the data collected from all mobile locations (normal, medium, high speeds). We note that the dotted line is closer to the indoor CDF. This leads to superimposed points located at the beginning of the tails of both the CDFs. These overlapped points are coming mainly from either deep indoor

positions or high or medium speed mobile positions, thus, creating ambiguity.

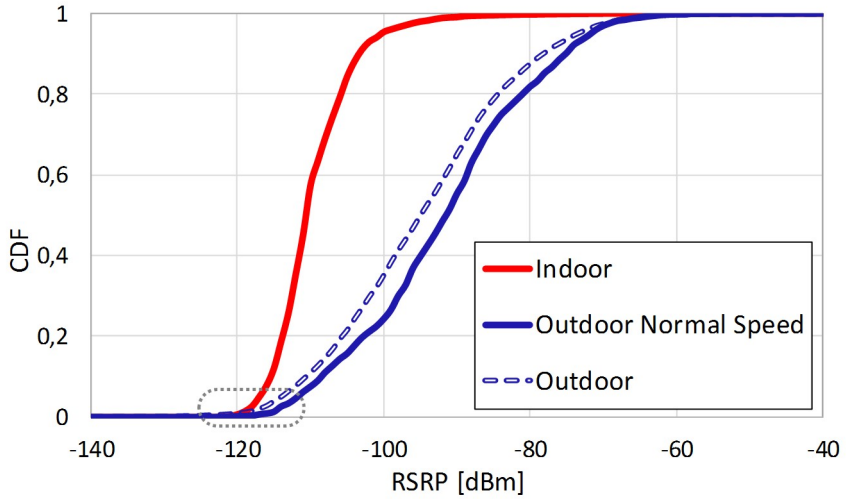

Figure 4.3 – Empirical CDFs for measured $RSRP$. (full) outdoor static and low speed, (dotted) outdoor variable speed.

Figure 4.4 shows the CDF of mobility indicator values for different environments (car, pedestrian, buildings, train, mall, bus) for $TCR_{max} = 100s$. As expected, the curves imply that the number of cell ID changes ($NCID$) is correlated with the environment type. Indeed, the indoor user (e.g. in buildings) either doesn't change cells or changes only a very few times when he is located at borders of multiple cells. However, when he moves outdoor (e.g in transportation), the number of handovers increases as he covers large distances. Note that the figure implies that $NCID$ is smaller in pedestrian case than in mall case. One reason is that in some cases relatively longer walks occur in malls.

Consequently, the addition of TA and MI features should eliminate the ambiguities as they provide additional information that is relevant to the IOD system. Indeed, with them the IOD system exploits information on the distance of mobile users from the base station and is aware of their mobility type, respectively. On one hand, TA would help to classify the ambiguous points which

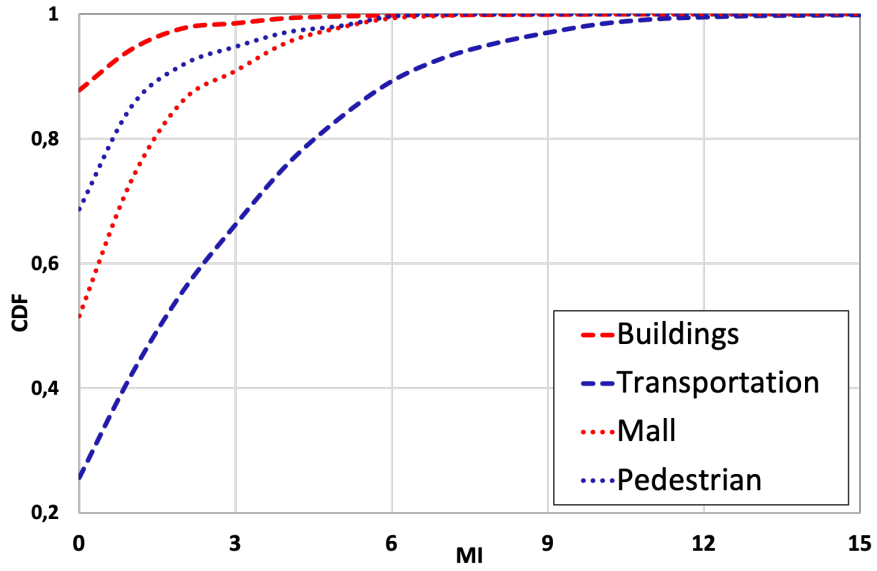

Figure 4.4 – Cumulated Distribution Function of the mobility indicator vs. Environment - one user

correspond to, for e.g., measurement points with low $RSRP$, but near to eNB. On the other hand, the user mobility highly correlated to the user environment will ease the classification of outdoor measurement points with low $RSRP$, for example, while inside a high speed train. The indoor user moves slowly as compared to outdoor where he can move more quickly. Therefore, using both the additional signals can help to classify the ambiguous measurement points and would improve the overall performance of the supervised classifier. However, how much does each parameter contribute to IOD performance?

Figure 4.5 depicts the relative optimal ordering of the four input features related to their relevance for IOD. The cumulative information brought by these features is equal to 100% (presented by the red line in the figure 4.5). The order of these features is obtained using "Extra-Trees-Classifier" algorithm. It reveals that $RSRP$ will contribute most to the IOD performance. Alone, $RSRP$ brought around 43% of information to the classifier. The ranking scores of TA and MI are close, bringing both around 25% of information to the classifier.

They thus impact the IOD performance almost identically. Furthermore, both signals together will contribute a little higher than $RSRP$ and CQI combined. Thus, an improvement in IOD performance is expected by introducing these two additional parameters.

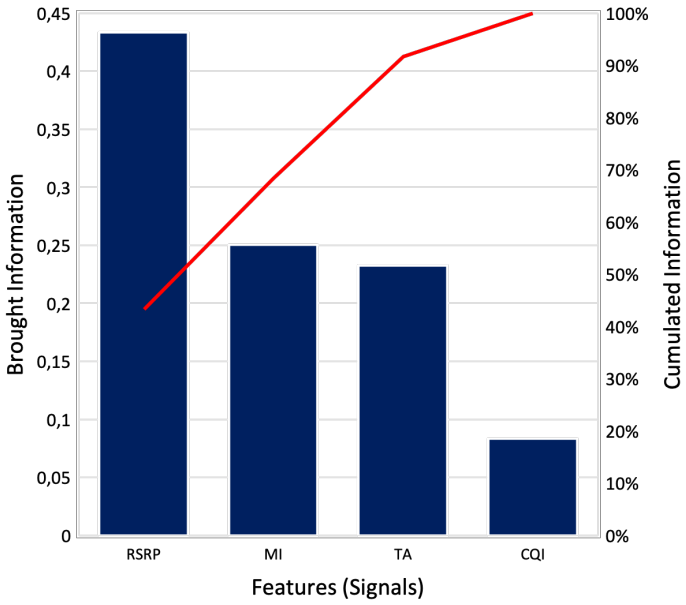

Figure 4.5 – Feature ranking based on cumulative information brought by them

4.3.2 Data collection mode: crowd-sourcing vs. drive-test

The dataset described in the previous chapter 3.3 has been collected using the crowd-sourcing mode. This mode is rarely used by operators. In crowd-sourcing mode, the collected data consists of signals measured by the mobile phone and sent to the eNB. It enables to better reflect the users' behaviors by capturing a huge diversity of their experiences, unlike drive test collection mode which imposes measurements in limited situations, generates data with a poor representation of a daily life of a mobile user.

Figure 4.6 shows the empirical cumulative distribution functions (CDFs) of $RSRP$ and CQI obtained with the dataset in the indoor and outdoor cases.

The significant offset between the indoor and the outdoor curves, results from substantial difference and attenuation variation in radio signal propagation. It is mainly due to reflection, diffraction, dispersion and attenuation experienced in indoor environment. However, we note that there is some overlap between the ranges of $RSRP$ and CQI values. Also the extreme values seen in the two indoor and outdoor CDFs (located in tails) get similar and the division between the two gets blurred. The behaviour at the juncture of extreme values can be explained by the ambiguous characteristics of the environment when a user is at high speed (Train, car, etc.) or when he is in a semi indoor environments (like balconies, semi-open building, near a window, etc). We argue that these points are ambiguous and will pose a good challenge for supervised classification, since they can be indifferently classed indoor or outdoor at the same time.

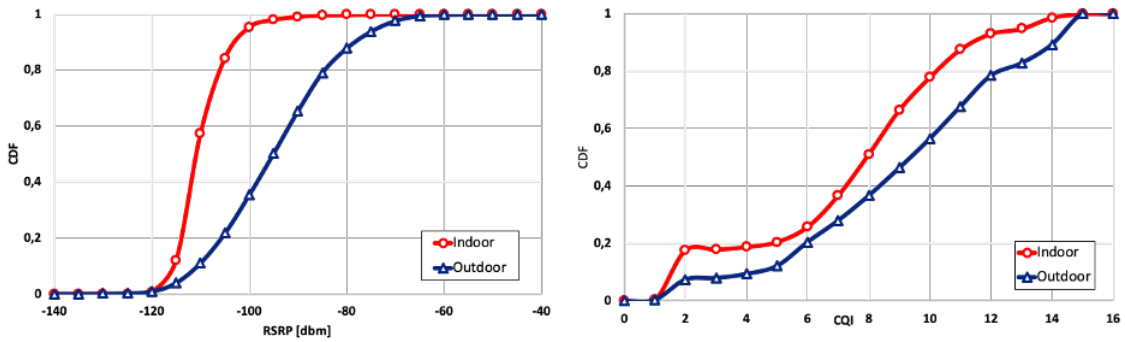

Figure 4.6 – Empirical CDF for measured RSRP (left) and CQI (right) in crowd-sourcing mode: multiple environments and places - Indoor (red) and Outddor (blue).

An alternate data collection mode, which is widely used by operators to collect data, is known as the drive-test mode. However, this mode imposes limits on capturing the reality through the data collected. Such data collection campaigns are run for limited hours per day during short periods (couple of weeks) and at some specific places. To model this way of collecting data, we extracted a portion of data (EPD) from the whole dataset. By this way to select EPD data, we aimed to be as close as possible to the type of places where some drive-tests

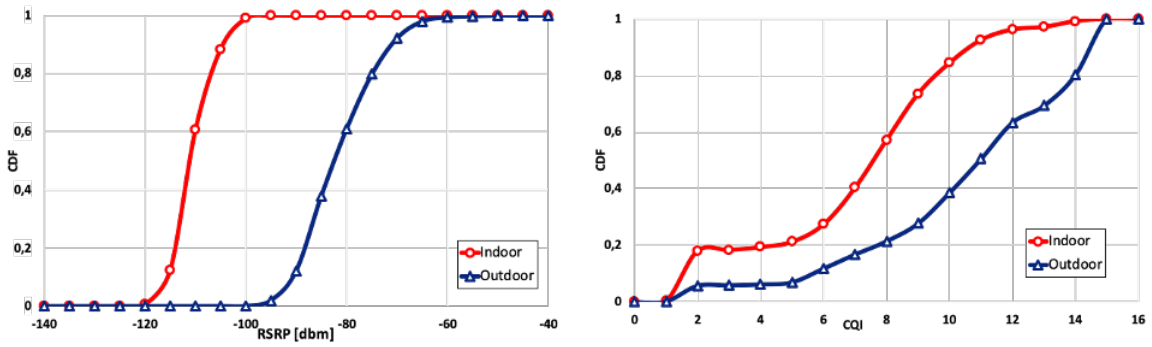

Figure 4.7 – Empirical CDF for measured RSRP (left) and CQI (right) in drive-test type mode: specific environments and places - Indoor (red) and Outddor (blue).

were performed by one of the top 3 American operators in New York City in [21]. Therefore, to build EPD we only considered data captured in metropolis (Paris and southern suburbs see figure 4.8). Indeed, Paris as metropolis, has a dense and a specific architecture which allows better comparison with NYC. Concerning indoor data, we selected instances where the user was strictly indoor and, thus, not in "semi-indoor" positions like semi-open building or balconies, etc. For outdoor data, we chose the instances where the user was either pedestrian or in vehicle in different city streets (limited speed). Thus, to mimic drive-tests, we consequently ignored data coming from environments like subway, countryside, forest, beaches, mountains, etc. We did this to enable a fair comparison between the two modes.

Figure 4.7 shows well separated $RSRP$ empirical cdfs between the indoor and outdoor classes. Note that the superimposed points of both the cdfs that we previously judged as conflicting have disappeared. The overlap between both the cdfs, which previously led to ambiguity, has disappeared. This is due to the significant distance between the indoor and the outdoor curves. If we draw a vertical line at -100 dB in both $RSRP$ CDFs, in the case of EPD (Fig 4.7) we can easily notice that there are almost no outdoor measurements below -100 dB. However in the case of crowd-sourcing data (figure 4.6) there are around 30%

of measurements below -100 dB. Similar phenomenon is noticed in the case of CQI cdfs. This analysis allows us to argue that supervised classification will run better on labeled dataset collected in drive-test mode as compared to obtained through crowd-sourcing mode because there are some thresholds that will help to map the features to the desired output (IOD).

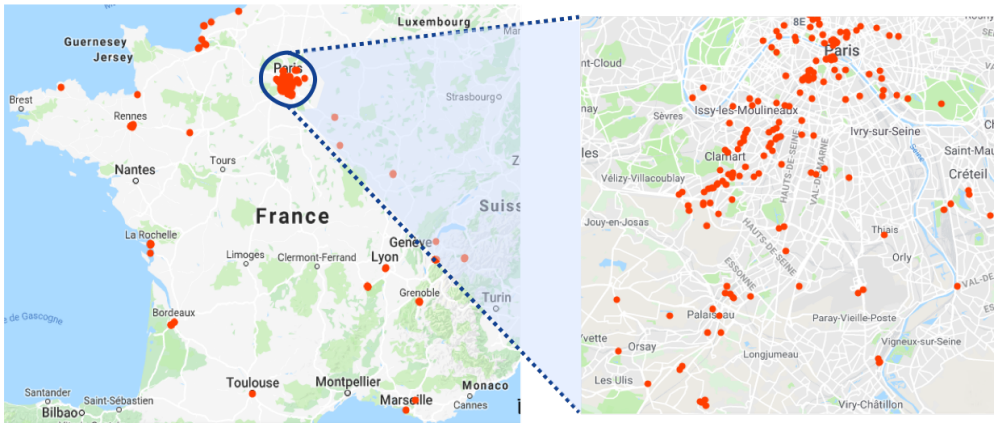

Figure 4.8 – The Data collection Points of EPD in drive-test like mode: Paris and southern suburbs

The evaluation of both input pairs ($RSRP$, $RSRQ$), which is the reference input for IOD in the literature, is conducted in three specific cases, that enable to make a comparison between drive-test mode and crowd-sourcing mode:

— Training and evaluation on labeled EPD collected in drive-test like mode (see Tab. 4.1),

— Training on labeled EPD from dive-test mode and evaluation on labeled data collected in crowd-sourcing mode (see Tab. 4.2) and,

— Training and evaluation on labeled data collected in crowd-sourcing mode (see Tab. 4.3)

As shown in table 4.1, running either classification (SVM, Random Forest, Neural Network) or clustering (k-means), algorithms on labeled EPD, obtained from drive-test like mode, shows good performance with an $F1-score$ of more than 99%, which is the reference result found in literature [21]. However, af-

ter training the module using labeled EPD, when we use it to perform IOD classification on crowd-sourced data then it leads to a dramatic performance deterioration, as shown in table 4.2. $F1-score$ drops to 36.13% with SVM. The best performing algorithm is RandomForest, which gives an $F1-score$ of 61.99%. But, this is not an acceptable performance for IOD. The performance of a supervised classifier, in the third case, with training as well as evaluation performed on the crowd-sourced labeled data, is shown only for the case of SVM, which is the best among the tested algorithms if IOD is done with the pairs of data input ($RSRP$, CQI). Table 4.3 shows an noticeable enhancement of $F1-score$ to 83.66%.

Algorithm	RSRP-RSRQ		RSRP-CQI	
	Accuracy	F1-Score	Accuracy	F1-score
k–means	$99,68\%$	99.48%	99.67%	99.47%
SVM	99.75%	99.59%	99.76%	99.60%
NeuronalNetwork	99.50%	99.18%	99.57%	99.28%
RandomForest	99.83%	99.72%	99.77%	99.62%

Table 4.1 – Clustering and Classification performance: training and evaluation on labeled data (EPD) of drive-test like mode

As we guessed before, the performance of IOD classification when trained only on EPD and then tested on the crowd-sourced data drops in terms of F1-score and accuracy. This drop in performance is due to the presence of ambiguous points, combined with unknown environments frequented crossed by users in real life, which are not included in the drive-test data. Even when we know indoor or outdoor labels and the exact location of the mobile user, the knowledge obtained from EPD is not sufficient for a satisfactory supervised training as well as for classifying ambiguous points. Learning the user environment, only based on drive-test data, is thus not enough to learn the complexities of users' real life. Furthermore, in addition to the issue of ambiguous points, another problem surfaced when we closely observed our crowd-sourced

data. Indeed, we are facing the problem of imbalanced classes with more users indoor (around 70% in our case) than outdoor, which corresponds to the real user behaviour that the users are more likely to be indoor than outdoor, in their daily habits. This behavior also increases the challenge to do supervised training because of the imbalanced nature of the data.

Algorithm	RSRP-RSRQ		RSRP-CQI	
	Accuracy	F1-Score	Accuracy	F1-score
k–means	61.41%	60.07%	59.64%	57.77%
SVM	56.56%	36.13%	62.69%	61.71%
NeuronalNetwork	50.90%	44.58%	62.55%	61.54%
RandomForest	62.93%	61.99%	62.63%	61.59%

Table 4.2 – Clustering and Classification performance: training on EPD and evaluation on labeled data of crowd-sourcing mode

Algorithm	RSRP-RSRQ		RSRP-CQI	
	Accuracy	F1-Score	Accuracy	F1-score
SVM	85.48%	83.66%	85.54%	83.71%

Table 4.3 – SVM performance: training and evaluation on labeled data of crowd-sourcing mode

4.3.3 Performances of IOD

Architecture and configuration

As described in the table 4.9 the used DL with $(RSRP, CQI, TA, MI)$ module is a feed forward neuronal network (fully connected) with 8 hidden Layers using $tanh$ as the activation function. Actually, $tanh$ (see chapter 3) is one of the widely used activation function while designing neural networks today. It is used mainly in classification tasks which will lead to faster training process and convergence. As for the last layer (the output layer), we used a sigmoid activation function to smooth the results since we look for a binary classification

either 0 or 1 (for indoor/ outdoor environments). Recall, the labeled data is 72k is. The labeled dataset is split into two subsets as follows: 70% for training, 30% for validation and test.

Results

Table 4.5 presents the performance results of a supervised multi-output classification for IOD given in 4.4.

In this section, we study the impact of adding CQI and two additional input features referred to as MI and TA to existing $RSRP$ and on IOD performance. MI and TA represent respectively the mobility type and the distance between user and eNB.

The comparison of both input pairs ($RSRP$, $RSRQ$), which is the reference input for IOD in the literature, vs. ($RSRP$, CQI), has been conducted in three cases described in 4.3.2. As shown in tables 4.1, 4.2 and 4.3, running either classification (SVM, Random Forest, Neural Network) or clustering (k-means) algorithms shows good performance with higher $F1-score$ in most of the cases when learning model is trained with input data ($RSRP$, CQI). Comparing both tables, Table 4.1 and Table 4.3, shows that using ($RSRP$, CQI) as input provides similar results to the case when the pair ($RSRP$, $RSRQ$) is used for classifying crowd-sourcing data. The results are even slightly better in case of table 4.2 with ($RSRP$, CQI). The best algorithm among the tested algorithms is SVM, which gives the highest $F1-score$ and Accuracy. Table 4.3 shows an noticeable enhancement of $F1-score$ to 83.71% when using ($RSRP$, CQI) with SVM. However, $F1-score$ of 83.71% is a moderately acceptable performance for IOD task that targets to help autonomic networks in their self-management. We are still far from the reference in the literature. This can be explained by the fact that we are dealing with different datasets collected in different modes. In what fol-

lows, we will work on improving this score to reach 95% of F1-score. As pointed out in chapter 3, the margin of tolerated error that we fix is of the order of 5% for network optimization purpose.

For the evaluation of TA and MI impact on performance of IOD, the comparison is done in three cases where different structures of learning datasets are examined, see table 4.4. The first one contains only $RSRP$ and CQI, the second one includes in addition the timing advance, but not the mobility indicator and the third one includes all data. As shown in the table 4.4, TA and MI added to $RSRP$ and CQI have enhanced the classical machine learning performance with up to 8% of gain, approximately. We also note that Deep Learning (DL) outperforms the other classical machine learning algorithms. This experimentation performed using the $S_{Labeled}$ data, provides a DL model with 95.30% of F1-score.

Algo	RSRP-CQI		RSRP-CQI-TA		RSRP-CQI-TA-MI	
	Acc.	F1-Sc.	Acc.	F1-Sc.	Acc.	F1-Sc.
kMeans	78.73%	75.81%	66.61%	45.93%	75.83%	67.38%
Logis. Regress.	84.63%	82.26%	87.59%	85.93%	89.67%	88.44%
SVM	85.54%	**83.71%**	90.17%	89.11%	92.32%	91.44%
DL	**85.60%**	83.66%	**93.45%**	**92.77%**	**95.72%**	**95.30%**

Table 4.4 – Clustering and supervised Classification performance: Accuracy & F1-score vs. Timing Advance & Mobility indicator

4.4 Semi-supervised Deep Learning-based classification: IOD

First lets also talk about the quantity of the data. Assuming that we have a sufficiently powerful learning algorithm, then one of the most reliable ways

to get better performance is to feed the algorithm with more data. Indeed, the quality of the model is generally constrained by the quality and the volume of the training data. Deep Learning (DL) and other modern nonlinear machine learning techniques get better with more data. Thus, there is a need to look for a way to enlarge volume of the training data. The idea is then to use unlabeled data, which is easy to obtain, and mix it with available labeled data, which is costly to obtain, for classifier training. Hybrid and semi-supervised approaches are the best candidates for this.

In this section, we compare and discuss the IOD performance using semi-supervised IOD approaches. As investigated in [101], [102], [103], [104] and [88], we consider the following 3 classic approaches of hybrid learning that make use of both the labeled and the unlabeled data:

1. Cluster-then-label: a clustering method is used to label the unlabeled data.
2. Co-training: multiple supervised classifiers learn from each other's outputs.
3. Self-training: a supervised classifier trained on a small labeled dataset learns iteratively from its own classification of additional unlabeled data.

We evaluate the above 3 methods of learning using our dataset. Let S_{Total} be the total dataset made of:

$$S_{Total} = S_{Labeled} \cup S_{Unlabeled}$$

where $S_{Labeled} \in \mathbb{R}^6$ is the subset of the labeled data and $S_{Unlabeled} \in \mathbb{R}^5$ is the subset of the unlabeled data. Note that $\text{Card}\,(S_{Unlabeled}) \approx 3 \times \text{Card}\,(S_{Labeled})$ in case of our collected data. For the performance evaluation on new environments unknown to the classifier we use $S_{Test} \in \mathbb{R}^6$, where $S_{Test} \notin S_{Total}$ and $\text{Card}\,(S_{Test}) \approx 3 \times \text{Card}\,(S_{Labeled})$.

4.4.1 Cluster-then-Label

Our proposed system of Cluster Then Label (CTL) approach is composed of two main modules described in Figure 4.9. The first module handles the unlabeled data by applying a clustering algorithm on S_{Total} to make emerge 2 clusters: indoor and outdoor. We use labels of $S_{Labeled}$, as well as the apriori information that users are much more indoor than outdoor, to associate labels to $S_{Unlabeled}$. Then an optimizer is used to correct the wrong labels, as much as it can, during the clustering phase. For correction, we use the idea that a user can not change his environment twice in 30 seconds. The idea can be explained from the following example. Imagine that we have three consecutive points in the dataset which are very near in time. If, for example, the first point is mapped as indoor, the second is mapped as outdoor and the next point, very near in time, is again mapped as indoor, then we assume that there is an error in mapping. This is because a user cannot change its environment two times so quickly. The second module uses $S_{Labeled}$ and also newly labeled data of $S_{Unlabeled}$ to train a supervised classifier.

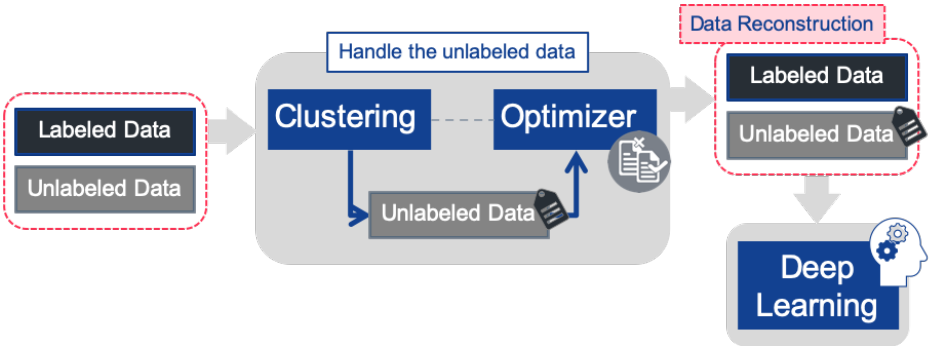

Figure 4.9 – The Cluster-then-label semi supervised approach model

We evaluate the CTL method with different clustering algorithms (K-Means, Expectation Maximization, Hierarchical Clustering, Bayesian Gaussian Mixture (BGM)). BGM showed better performance and was retained. Deep feed forward neural Network (DL) was used as the supervised classifier.

4.4.2 Co-Training

In general, the Co-training (CT) approach explores the results of two or more classifiers at the same time. There are many implementations of the CT according to the needs and the use cases. However, the most common one splits the dataset vertically according to features (signals in our case) and thus forming feature-based sub-datasets. As shown in Figure 4.10, two DL classifiers are trained on $S_{Labeled}$ data. Then each data instance in $S_{Unlabeled}$ is classified by the two classifiers and the intersection result with high classification probability is used to retrain and improve a final DL model. The assumption is that the classifiers working with different sets of features are able to complement each other. The main issue of CT is how to split data vertically to form the subsets that have the same amount of information. The idea is that if there are 2 primary features and 2 other secondary ones, then we will build subsets in a way that each subset has a primary feature and a secondary one. This guarantees that each subset and the associated classifier has its fair share of effective features to attain good performance.

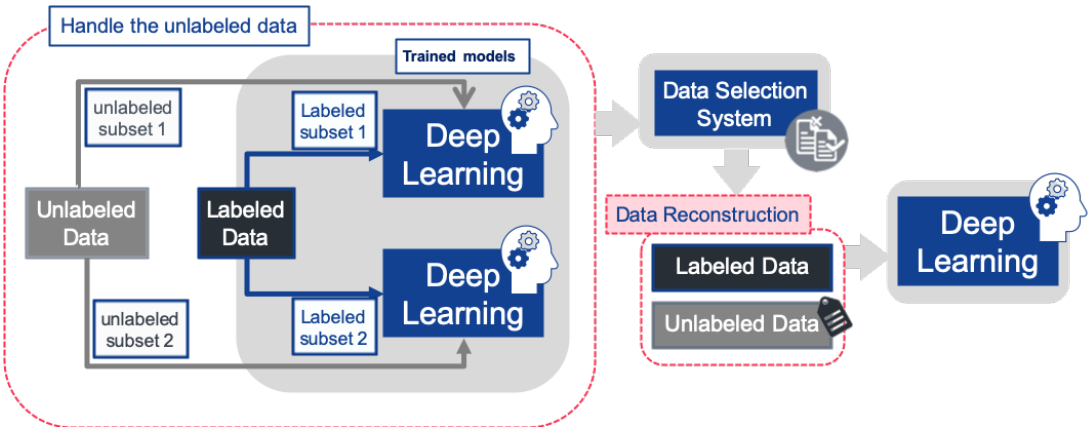

Figure 4.10 – The Co-Training (CT) semi supervised approach model

We analyzed the features in consideration with different machine learning techniques. According to Figure 4.5, we divided the whole dataset vertically

into two subsets composed of (1) [$RSRP, CQI, Class$] and (2) [$TA, MI, Class$]. This vertical division ensures the same information weight ($\simeq 50\%$), so we offer a fair opportunity of equal learning to both the DL classifiers. After the training phase of these two classifiers, we apply the same vertical division on the $S_{Unlabeled}$ set in order to predict their labels. Each data instance is classified by the two different classifiers. For the final step, only the intersection between resulting labels of two classifiers is kept. The classified part of $S_{Unlabeled}$, which is kept, is then added to $S_{labeled}$. This new data set is then used to train and improve the final DL module.

4.4.3 Self-training

The Self-Training (ST) approach is one of the semi-supervised learning methods that alternatively repeats classifier training and labeling unlabeled data in training set. The main issue with ST approach is the amplification of error while labeling the unlabeled data. That means if the current trained classifier makes errors while classifying the unlabeled data then the wrong label of the unlabeled data will provide an inaccurate information for the classifier of the next step [105]. By iterating these two steps, the overall error of the final classifier will become larger. To remedy this error amplification phenomenon and to have a generic classifier, we propose a data selection system between the two phases (Figure 4.11). To eliminate the wrongly labeled data, we again apply the assumption that a user can not change his environment twice in 30 seconds. A label is therefore considered wrong if in 30s the user goes from environment 1 to environment 2 and then from environment 2 to environment 1 again. Thus, we eliminate this labeling error. We also delete data that was classified with a low classification probability. That means, we eliminate data that was classified with a classification probability lower than 65%. This threshold of 65% was fixed after a statistical study to avoid both risks of over-fitting or error amplification.

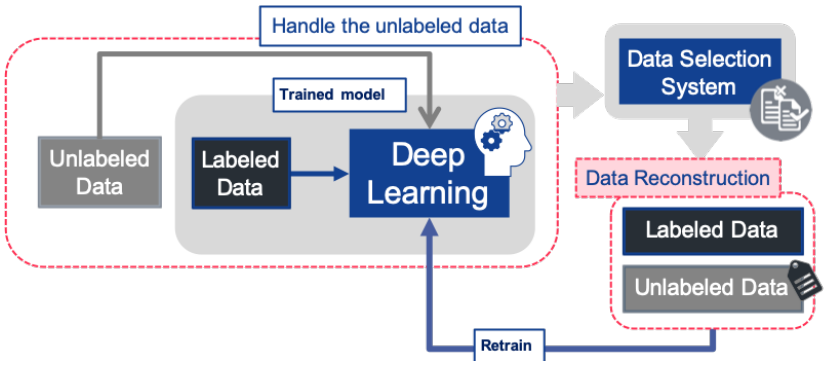

Figure 4.11 – The Self-taught/Self training semi supervised approach model

4.4.4 Performances of IOD

Architecture and configuration

For the semi-supervised study, we have used the whole dataset at our disposal at the time of the experiment. That is to say 250k measurements with 30% labeled and 70% unlabeled points. The used DL module by the 3 hybrid learning systems (CTL, CT and ST) is depicted in the section dedicated to evaluate the performance obtained with supervised learning 4.3.3. It is a feed forward neural network (fully connected) that has the same configuration as the model DL $(RSRP, CQI, TA, MI)$ described in the table 4.9. The experimentation performed on the $S_{Labeled}$ data, provides a $F1 - score$ of 95.30%. This model is saved and will serve as an initialization for the next training steps.

Results

Table 4.5 presents the performance results of a semi-supervised multi-output classification for IOD.

The 3 learning approaches - CTL, CT and ST - are evaluated on S_{Total} by computing each-time the F1-score and the accuracy of each approach. The performance evaluation is carried out on both S_{Test} and $S_{Labeled}$ data. As shown

in table 4.5, CTL has the lowest F1-score compared to CT and ST. This is explained by the fact that more the unlabeled data volume increases, the more the performance of the supervised DL gets limited to the clustering performance and errors. In our case, the BGM cluster used has a F1-score of 79.71%, which gives a low F1-score of CTL. CT and ST show close performances with slightly better performance of the latter since both the DL classifiers trained with their own tagged data subsets provide the same F1-score, with an average of 85%. However, CT is very complex and greedy in resource use. CT takes lot of training time as it deals with 3 neural networks. Therefore, ST is the best choice for IOD system, since, on the first phase (trained only on labeled data) as well as the last phase (trained on both labeled and unlabeled data), it has shown the best performances reaching an $F1 - score$ of 96.18%.

CTL		CT		ST	
Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-score
83.05%	79.89%	95.77%	95.34%	**96.50%**	**96.18%**

Table 4.5 – Semi-Supervised approach (CTL, CT, ST) performances: Accuracy & F1-score

4.5 What if more granularity of UED is taken into consideration?

In this section, we investigate the user environment detection (UED) considering more than two types of environments. Indeed an environment described with a more thinner granularity reflects more the complexity of a user's daily life and captures better the variety of his movements in real world. However, increasing the number of classes, that is required to describe the diversity of the environment, generates groups of data instances that are distributed

unequally between the different categories. The unequal distribution of data constitutes an issue for ML.

Problematic 5

A model is required to classify a user's environment to what granularity level? Which classification scheme can tackle the issue of imbalanced data? Can we classify the user environment with detailed classes, with good performance?

Contribution 5

In the following, a comparative analysis of classification between schemes of three, four and five classes will show that using four classes with relevant labels enables a good trade-off between balancing the dataset and more granularity.

We investigate eight classes of environments that are typical of the environments frequented by the user during his or her daily life: *Work*, *Home*, *Building*, *Bus*, *Car*, *Mall*, *Pedestrian*, *Train*. The detection is done using a supervised as well as a semi-supervised, multi-output, classification techniques. Multi-output refers to the detection of multiple types of environments. We shall look for a best trade-off between more granularity and performances, namely for the highest number of classes available and an $F1-score$ higher or equal than 95%.

4.5.1 Relation between user activity and environment type

UED task is performed using the same data described in section 4.2 of this chapter. That is to say, we are using our dataset composed of radio signals ($RSRP$, CQI), temporal features (TA, $Time$), a Mobility Indicator (MI) derived

from Cell ID changes $NCID$, and finally the environment. The only difference is that the time value will also be exploited for more than optimization purpose since there is a high correlation between the time and the user environment.

Obviously, the environment class is no more just binary (Indoor/ Outdoor). We consider the following categories of environment: [*Work*, *Home*, *Building*, *Bus*, *Car*, *Mall*, *Pedestrian*, *Train*]. They have been chosen for labelling the data since they are the common places most frequented by the users in our dataset. In other word, they are highly representative of the locations where data has been collected. These eight environment categories were selected in order to reflect the complexity of a user's daily life and capture the variety of his movements in real world. As explained and detailed in chapter 1, people spend most of their time indoor than in mobility and outdoor. Consequently, outdoor labels are less represented than indoor labels in phone usage data.

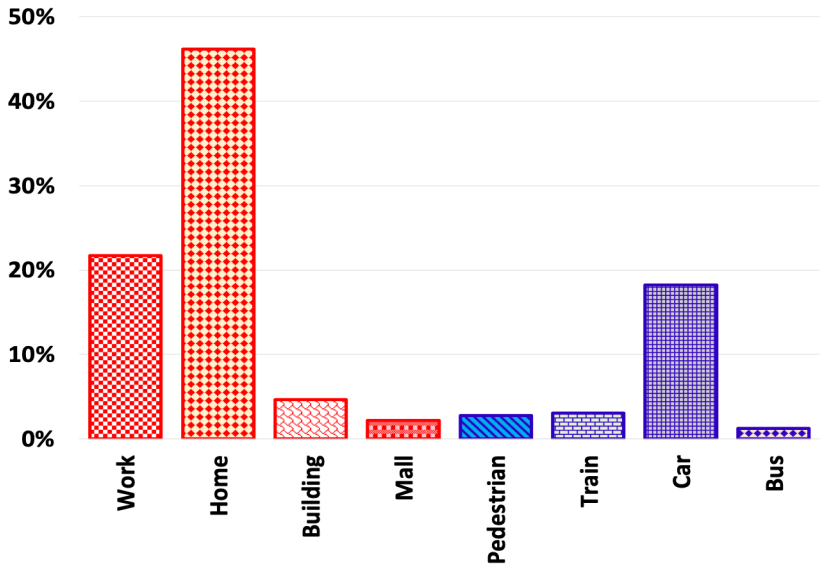

Figure 4.12 – User activity per environment

To analyse the user behavior, using the collected labeled data, we fix the definition of the 'phone usage' state. The 'phone usage' state is defined as the state when the user uses his phone or equivalently the screen is on, unlocked,

and there is data exchanged. In order to track this state, following additional data features are also collected during the campaign. These features are the instance index or timestamp, the screen state and the label of environment type. Analysing this data gives us a clear picture of mobile phone usage in different environments.

Figure 4.12 depicts the user activity by plotting the phone usage ratio for different environment categories. User activity is measured as the ratio between the number of instances the user has been using his phone effectively and the total number of instances. This figure illustrates the percentage of total time the user is connected to 4G network and exchanges data with it. We observe that most of the activity is spent indoor (70%), mainly at home and at work, as compared to being outdoor. It highlights the user activity trends that we observed after statistical analysis on mobile user behavior in literature. Figure 4.12 illustrates the multiple diversity of indoor and outdoor situations experienced by the users of mobile phone. These situations are [*Work*, *Home*, *Building*, *Bus*, *Car*, *Mall*, *Pedestrian*, *Train*]. Consequently, it is preferable to consider the user environment detection task as a multi-output classification problem.

We also observe in Figure 4.12 that the data instances are distributed unequally between the different categories. It shows that the data proportion in groups of label “Train”, “Bus”, “Mall”, “Building” or “Pedestrian” is very low compared to the other groups. This imbalanced nature of data remains a challenge for Machine Learning algorithms. This nature of data is in fact inherent to the user behavior: people are mostly in static conditions than in high speed. Moreover, as soon as the user speed increases, the network quality deteriorates and consequently the user switches to 3G or even to GSM (sometimes).

Another factor that also deeply affects the consumption of a mobile service in a given environment is the time. Figure Fig.4.13 shows the phone usage over

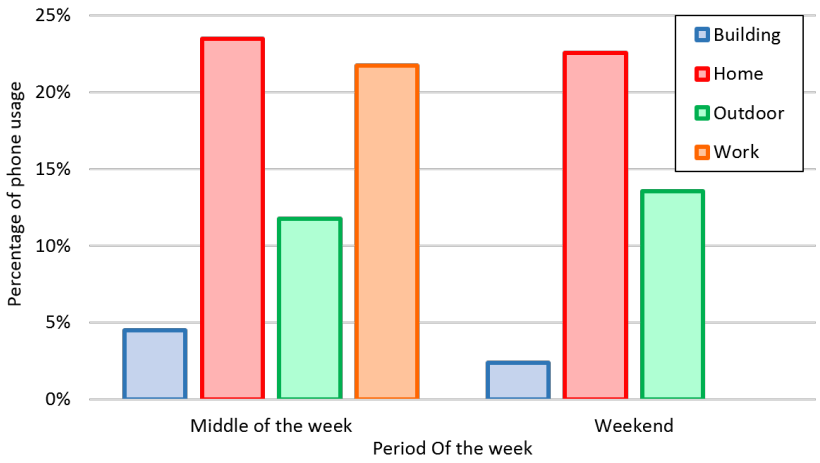

Figure 4.13 – User activity per environment during week period vs. weekends. Considered Environments: Building (Malls and other buildings type), Home, work, Outdoor(Car, Bus, Train, Pedestrian)

the period of a week and puts the emphasis on differences between weekdays and weekends. We see that indoor environments are a major stage for most of the users' phone activity, on weekend as well as the during week days. Although, regardless of the point in time during the week, indoor class always keeps its importance in terms of mobile use, there are variations between different indoor environments when the period of the week is different. The work environment is totally absent on weekend, but has nearly the same weight as the home one.

By observing figure 4.14 the pedestrian, transport and mall environments, it is noted that user mobility becomes more important on weekends. These statistics also show the rise of the percentage of mobile use in outdoors, during weekend, which is justified by the fact that distraction activities reach their peaks on weekends. It is also to note that the weekend period holds a lot of surprises in terms of mobile use, because this period is subject to all kinds of human preferences which makes it very diverse. Analysis over the day (see figure 4.15) also shows that time is an important factor linked to user activity and user environment. In the example here, a normal user activity begins ap-

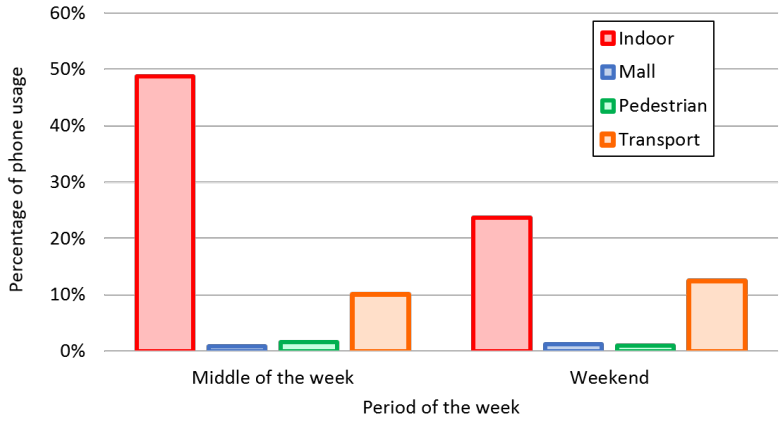

Figure 4.14 – User activity per environment during week period vs. weekends. Considered Environments: Indoor (Home, Work, and other buildings type), Pedestrian, transport (Car, Bus, Train)

proximately between 6 a.m. and 8 a.m. at indoors (the user is at home). Then, we have a peak between 9 a.m and 11 a.m of phone use at outdoors which can be explained by service consumption during transit to daily user's activity places: work, high school, university or other. A second peak of mobile activity in transport environments also occurs between 12 a.m and 2 p.m, which corresponds to lunch time, as it is normal for people to take transport to eat somewhere. As for the phone usage in mall environment, we can see, it usually starts in the afternoon from 3 p.m on wards because people generally prefer to go to the shopping centers in the afternoon and after work. The period between 2 a.m and 4 a.m is a hollow period where the phone use activity is almost zero because most people are sleeping.

As we can see, the time is deeply linked to the user activity in different environments. Thus, for the user environment detection, we also include time in our features for training. The dataset is composed of 6-feature samples as follows: $Time$, $RSRP$, CQI, TA, MI and the $Environment$ (this label corresponds to the environment traversed by user during the measurement campaign). We

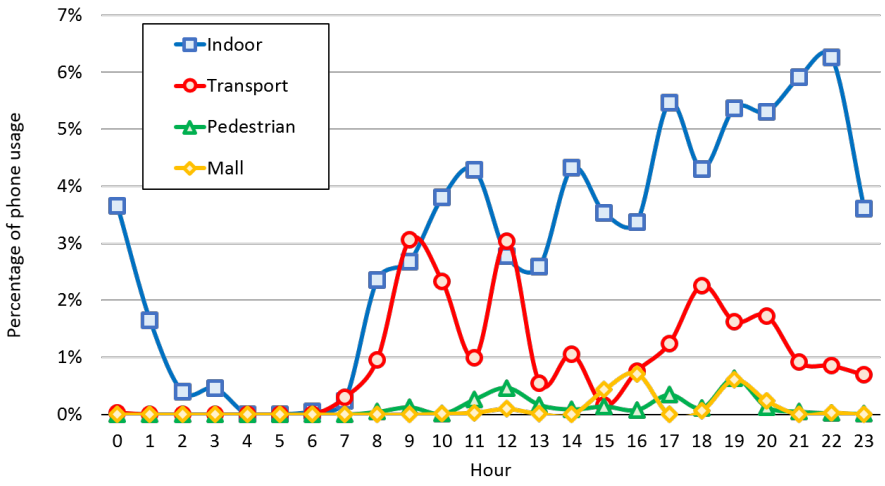

Figure 4.15 – User activity per environment during day hours. Considered Environments: Indoor (Home, Work, and other buildings type), Pedestrian, transport (Car, Bus, Train)

consider the following categories of environment: [*Work*, *Home*, *Building*, *Bus*, *Car*, *Mall*, *Pedestrian*, *Train*]. They have been chosen for labelling the data. This is because they are highly representative of the locations where data has been collected. The environment labeling was processed using GPS coordinates for users who allowed us to collect them.

4.5.2 Classification schemes

Distinguishing between specific outdoor environments or indoor ones that are rarely visited is a hard task for any ML algorithm because of the lack of data corresponding to them. Some environments are not often visited by individuals and not even periodically visited, what makes the classification task difficult for such type of environments. Aggregating rare environments, with thick relationships among them, into significant groups helps to resolve that issue. This is because combining them results into more consistent and recognizable groups. The only factor on which such grouping depends is the logical closeness between environments. We seek thus a compromise between a simple

binary classification problem that was efficiently performed in previous section of this chapter, and a complicated classification task that holds detailed information about the user's profile, but cannot be successfully accomplished given the available data on occasional environments.

Thus, we aim to define relevant classification schemes with multiple classes by smartly regrouping the various environment categories: [*Work*, *Home*, *Building*, *Bus*, *Car*, *Mall*, *Pedestrian*, *Train*]. Recall that the issue of imbalanced data is inherently, due to the nature of real human activities. As a consequence, in order to ensure an efficient scheme of environment classification for real user activity, we have to find a trade-off to limit this inherent data bias for different classes. We aim to design a classification scheme that detects the detailed environment types of mobile users, with a fine granularity and a low decision error.

We focus on studying the relevant trade-offs in our case of multi-output classification. We consider different possibilities ranging from a simple binary classification problem to a more complex classification task of detecting detailed information about a user's environment. To ensure this, we decide to regroup the environment categories {*Work*, *Home*, *Building*, *Mall*, *Bus*, *Car*, *Pedestrian*, *Train*}. Furthermore, these categories have thick inter-relationships and we can merge them into bigger consistent and recognizable groups. The created merged groups shall also include data with similar statistical properties in order to optimize the classification results. This is obtained by observing the cumulative distribution curves of the collected data $\{RSRP, CQI, TA, MI\}$ as well as the variance σ^2 of the phone activity for each proposed multi-class schemes.

The variance represents the percentage of total instances linked to this environment and is written as:

$$\sigma^2 = \frac{\sum(X_i - \bar{X})^2}{N}$$

where X_i is the number of instance in the class i and N the number of classes. The variance is an appropriate metric to measure the degree of imbalance by quantifying the variance of the percentage of total instances per environment category from its average.

Figures 4.16 and 4.17 depict the cumulative distribution curves of $RSRP$ and MI, respectively. We observed in Figure 4.16 that three groups of similar curves can be extracted: a set with only *Home*, a set regrouping {*Work*, *Building*, *Mall*} and another set assembling {*Bus*, *Car*, *Pedestrian*, *Train*}. This results into a further split of indoor and outdoor classes. Furthermore, the environment "Home" is detached from the set of other indoor situations.

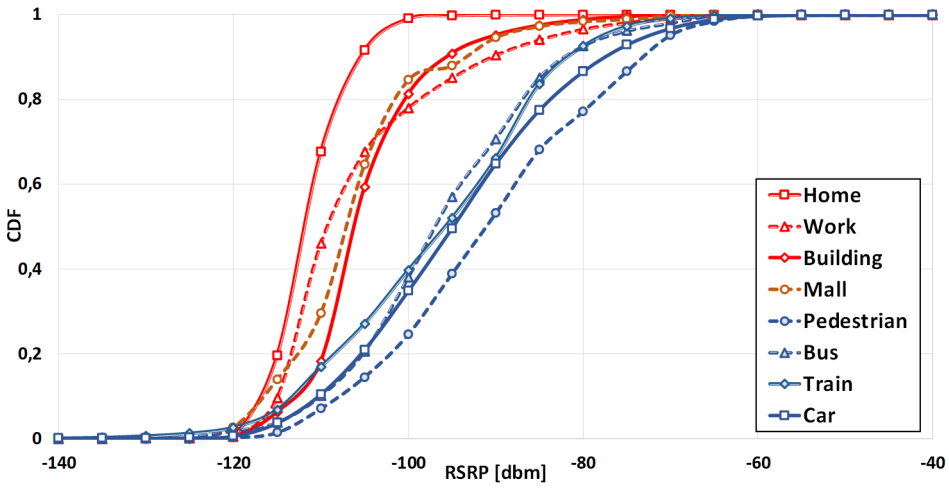

Figure 4.16 – CDF of RSRP

Analysing Figure 4.17, we also note a similar separation between indoor and outdoor curves. Moreover, CDF curves of MI highlight a clear separation between "pedestrian" and the remaining set of outdoor labels as well as between "Mall" and the others indoor situations. Clearly these two environment types are associated to users moving very slowly (walking) as compared to others that are either static or high speed.

Based on these observations, we propose to group the eight environments

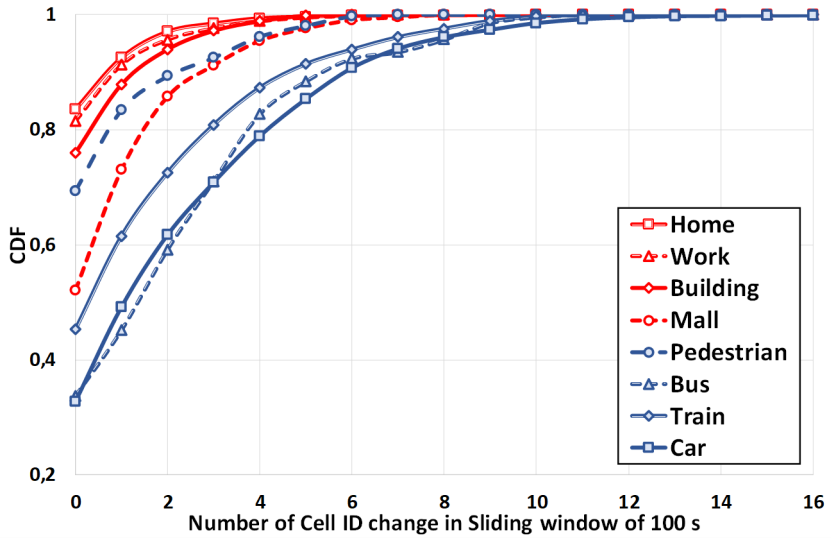

Figure 4.17 – CDF of number of Cell ID changes during 100s

as shown in Table 4.6. The label "Indoor" refers to "Home", "Work", "Mall" and "Buildings". The label "Outdoor" contains the environments "Pedestrian", "Car", "Bus" and "Train". "Transport" includes "Bus", "Train" and "Car". "Buildings" contains remaining indoor locations and various "Buildings". Thus, UED classification schemes investigated are given in Table 4.6:

UED scheme	Environment
$2C$	Outdoor, Indoor
$3C_0$	Outdoor, Buildings, Work
$3C_1$	Outdoor, Buildings, Home
$4CO_0$	Outdoor, Buildings, Work, Mall
$4CO_1$	Outdoor, Buildings, Home, Mall
$4CI_0$	Pedestrian, Transport, Buildings, Mall
$4CI_1$	Pedestrian, Transport, Buildings, Home
$5C_0$	Pedestrian, Transport, Buildings, Work, Mall
$5C_1$	Pedestrian, Transport, Buildings, Home, Mall
$8C$	Home, Work, Buildings, Mall, Pedestrian, Bus, Train, Car

Table 4.6 – UED classification schemes

For illustration, the examples of schemes “5C_0” and “5C_1” are provided in Figure 4.18.

Figure 4.19 shows the variance of data size versus each class schemes when the data is divided according to the above schemes. This is to quantify the balance or imbalance of data between different classes. When the variance is high, the dispersion, in terms of data size in different classes, is important and, thus, the scheme is very imbalanced. As shown in Figure 4.19, the schemes “3C_0” and “3C_1” have the smallest variance. They are followed by “4CO_1”, “4CI_1” and “5C_1”. Indeed, the scheme 3C results in the most balanced data among different classes. This scheme regroups the instances coming from outdoor labels and splits the indoor label instances.

All the schemes are illustrated in Figure 4.20. The figures represents the mobile phone activity according the environment type for the 8 proposed schemes. Precisely, the figure indicates the percentage of time a user is active on his or her mobile phone. Precisely, the figure indicates the percentage of time a user is active in his or her phone.

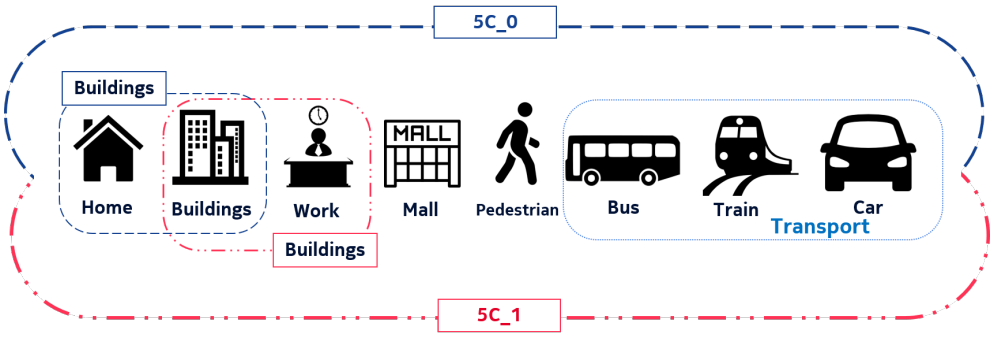

Figure 4.18 – Multiple class schemes example: “5C_0” and “5C_1”

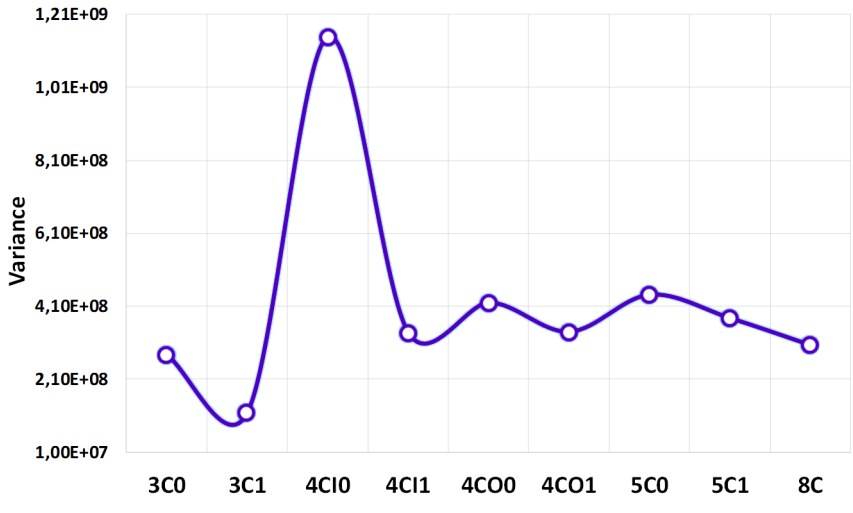

Figure 4.19 – Variance of phone activity for the multiple class schemes

4.5.3 Performances of UED

Architecture and configuration

For this experiment, we have used 270K lines of data collected only in France and corresponding to LTE network. The dataset is made of 50% of labeled data (for the eight environments) and 50% of unlabelled data. The training is done using the labeled part of our dataset. For training, we used (70%) of the labeled data and we used the (30%) remaining for evaluating the model's performance.

The set of hyperparameters (e.g. the number of hidden layers, batch size, epoch size, the weights) have been tuned using Bayesian optimization. The implementation is done under python and using Keras with Tensorflow as a back-end (we summarize them in the table 4.9).

For our supervised multi-output classification problem, we used a Feed Forward Neural Network architecture with a total of 7 hidden layers. To evaluate the impact of adding more data during the training phase, we used a semi-supervised Self-Training training model, (figure 4.11) detailed in section 4.4,

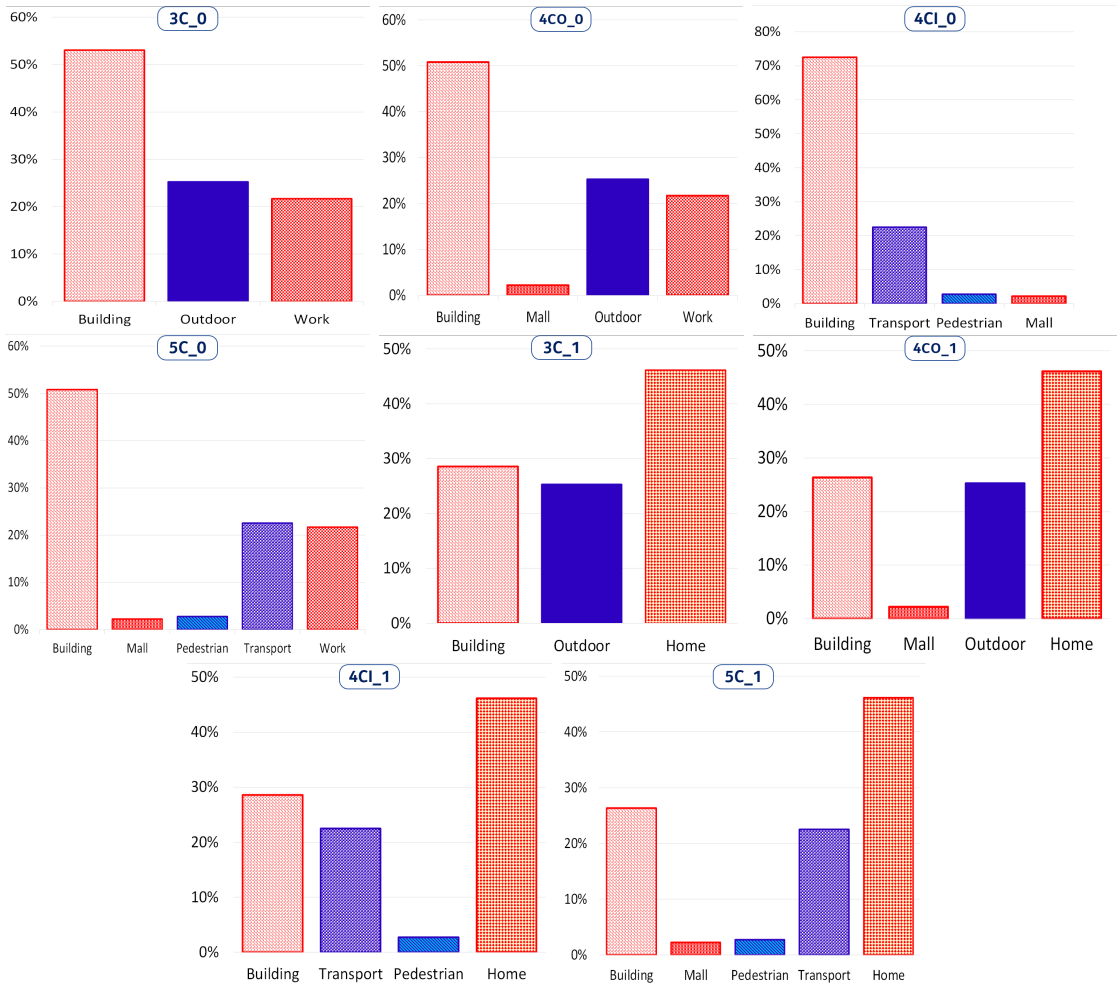

Figure 4.20 – Phone Activity for various classification schemes of environment

	2 Classes		3 Classes					
	$2C$		$3C_0$		$3C_1$			
	Acc.	F1-Sc.	Acc.	F1-Sc.	Acc.	F1-Sc.		
Sp.	**95.80**%	**95.76**%	91.89%	91.84%	92.06%	92.04%		
S.Sp.	**96.51**%	**96.45**%	94.20%	94.19%	94.14%	94.13%		
	4 Classes							
	$4CO_0$		$4CO_1$		$4CI_0$		$4CI_1$	
	Acc.	F1-Sc.	Acc.	F1-Sc.	Acc.	F1-Sc.	Acc.	F1-Sc.
Sp.	91.58%	91.53%	91.42%	91.38%	**93.18**%	**93.11**%	**94.31**%	**94.28**%
S.Sp.	94.24%	94.24%	94.18%	94.17%	**94.92**%	**94.92**%	**94.53**%	**94.55**%
	5 Classes							
	$5C_0$		$5C_1$					
	Acc.	F1-Sc.	Acc.	F1-Sc.				
Sp.	90.70%	90.69%	90.79%	90.72%				
S.Sp.	93.42%	93.38%	93.66%	93.63%				

Table 4.7 – Deep Learning-based supervised and semi-supervised multi-ouput classification performance: F1-score vs. classification schemes

since it showed the best performances among the 3 tested approaches. We presented these results in [41].

Results

Table 4.7 presents the performance results of both a supervised and a semi-supervised multi-output classification given in table 4.6 and with the set of hyperparameters of the table 4.9.

They have been obtained for the 8 multi-output classification schemes as well as the two-class scheme (IOD). They are evaluated in terms of $F1-score$ and then compared with the classical IOD binary classification. We observe that all schemes deliver $F1-scores$ higher than 90% that correspond to acceptable performance in terms of classification. However, among all the schemes, the two-class scheme and the four-class schemes “4CI_1” and “4CI_0” give the best performance.

As shown in Table 4.7, the F1-score of scheme "2C" is equal to 95.76%, when supervised training is used. The F1-scores of "4CI_1" and "4CI_0" obtained in supervised case are equal to 94.28% and 93.11%, respectively. We notice a slight improvement of F1-score using the two-class scheme as compared to the F1-score obtained under same conditions in [40]. Indeed, this enhancement is due to the fact that now our algorithms process two times more real radio data collected in real conditions of crowd-sourcing. Consequently, it impacts the performance in a favorable way. Using the semi-supervised method for multi-output detection is furthermore positive. It still enhances the scores thanks to the addition of unlabelled data in the training phase. F1-scores for the three schemes are equal to 96.45%, 94.55% and 94.92%, respectively. In both methods, we also observe a maximum loss of around 6% when using a five-class scheme as compared to the binary classification. The loss is reduced to around 2% when using a 4-class scheme.

Thus, we show that a detailed learning of the environment can be achieved with a very minimal loss of performance. This is obtained using a smart division of the groups "indoor" and "outdoor" into different sub-groups. We observe that the coarse learning, of the environment, benefits more from the diversity brought by the introduction of labels "pedestrian" and "in transport" than by the split of the group "indoor". Furthermore, we also note that the level of imbalance in data has an influence on the F1-score. The most imbalanced scheme delivers small F1-scores. The two best schemes (outside the two-class scheme) offer relatively balanced classes.

4.6 Labelled data volume vs. unlabelled data volume

In this section, we investigate the impact of the volume of labeled data or unlabeled data used on $F1-score$ of IOD.

Semi-supervised learning has be shown in preceding section as an efficient solution to handle the tagging data issue while guaranteeing better performance for UED or IOD. However we can also envisage semi-supervised learning as a solution to minimize the amount of tagging data issue while guaranteeing satisfactory performance for the task of environment detection. Indeed, as shown in table 4.5, with Self-Training system for IOD, we obtain a $F1-score$ of 96.18% by processing $72,7\%$ of unlabeled points and $27,3\%$ of labeled data. We observe that the attained $F1-score$ is 1% more than the target threshold, we imposed, to qualify a satisfactory performance of a task targeting to contribute positively to network optimization.

So, here, we address an important question: how much percentage of labeled data is needed to target a satisfactory performance for the task of environment detection system? Such an answer is of interest to operators. This is because, during online labeling phase, limiting the amount of data to label alleviates network overload by limiting the amount of UL signalling (all labels) sent to eNB. Additionally, it reduces the complexity and the required time for tagging data. Thus, the idea is to use the available labeled data, which is costly to obtain, and combine it with unlabeled data, which is easy to obtain, for classifier training. The answer to this question highly depends on the dataset and is specific to the use case. It is not a generic answer that can be generalized to other studies.

To answer this question, a study is conducted in the case of semi-supervised classification of IOD using the hybrid Self-Training (ST) system. For this, $F1-score$ is assessed while varying either the volume of labeled data or either the

volume of unlabeled data during the model training. For this, $F1-score$ is evaluated for various $S_{Unlabeled}$ and $S_{Labeled}$. Recall that S_{Total} represents the total dataset made of:

$$S_{Total} = S_{Labeled} \cup S_{Unlabeled}$$

where $S_{Labeled} \in \mathbb{R}^6$ is the subset of the labeled data and $S_{Unlabeled} \in \mathbb{R}^5$ is the subset of the unlabeled data. So, $F1-score$ is investigated leading to two scenarios:

— Scenario 1: $S_{Labeled}$ is fixed and the volume of $S_{Unlabeled}$ is varied. The ST training performance is evaluated progressively according to the percentage of unlabeled data which reaches a maximum of 72.69%.

— Scenario 2: $S_{Unlabeled}$ is fixed and the volume of the $S_{Labeled}$ is varied. The ST training performance is evaluated progressively according to the percentage of labeled data which reaches a maximum of 27.31%.

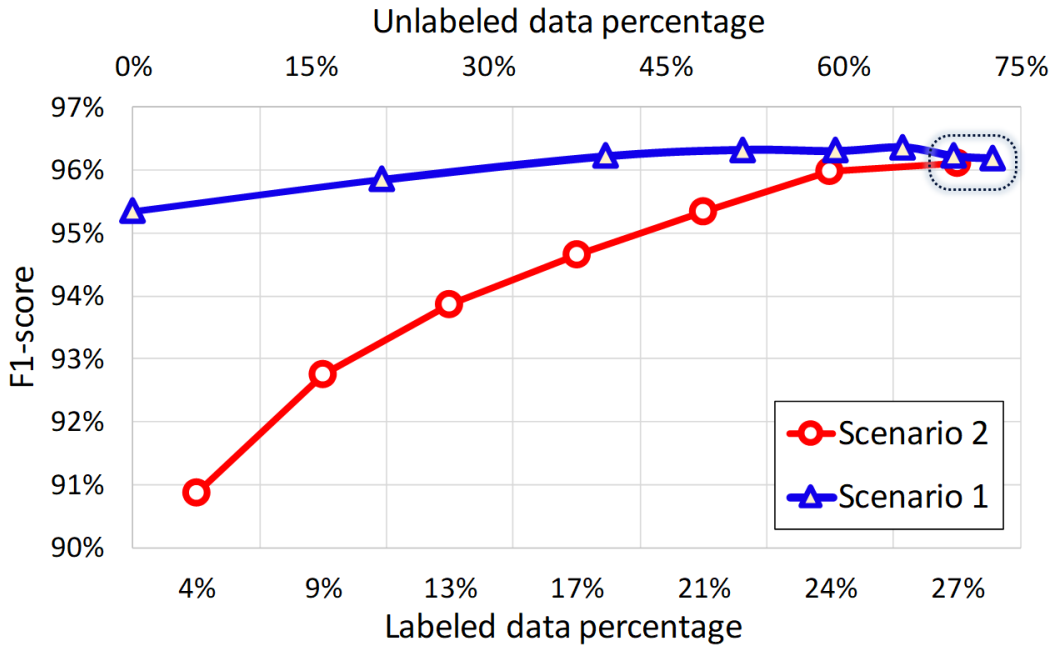

Figure 4.21 – Data volume Impact: (Blue Line) Scenario 1: Variation of the $S_{Unlabaled}$ volume. (Red Line) Scenario 2: Variation of $S_{Labeled}$ volume

Results of both scenarios are shown in Figure 4.21. The figure plots the F1-

score values of ST versus the size ratio between $S_{Labeled}$ or $S_{Unlabeled}$ and S_{Total}. The double X-axis refers then to the percentage of labeled data (the bottom X-axis) or unlabeled data (the top X-axis). The percentage is calculated from the total volume of data.

As expected, the addition of unlabeled data improves the IOD system performances. In scenario 1, ST uses all the labeled data and a variable part of unlabeled data. F1-score increases with the size of $S_{Unlabeled}$ data. However, there is only moderate improvement. By using all of the labeled data ST starts already at 95% to converge toward $96,18\%$ with a distribution of 27.31% and 72.69% of labeled and unlabeled data respectively. This state corresponds to the case where all the unlabeled data is used. The information brought by all $S_{Labeled}$ data is sufficiently rich.

This is unlike the scenario II, where the F1-score augmentation is more pronounced. Availability of less $S_{Labeled}$ is realistic assumption as collecting labeled data is expensive. In any case the labeled data contains more relevant information. If the mobile operator targets an error percentage of 5% for IOD (namely $F1 - score = 95\%$), the red curve indicates that a distribution of 19% and 81% of labeled and unlabeled data respectively is sufficient for the training phase. Consequently, the mobile network operators wanting to implement IOD inside their network may use similar percentages of labeled and unlabeled data during the updating phase of IOD learning model. They may need to manually label only 19% of collected data.

If 1% reduction of $F1 - score$ is accepted to reach 95%, it enables operators to gain 30% of signalling load for labeling. The table 4.8 provides for a target performance comprises between 90% and 95% the saved signaling load versus the target $F1 - score$.

Target $F1-score$	91%	92%	93%	94%	95%
Saved signaling percentage	85%	74%	67%	52%	30%

Table 4.8 – Percentage of labeled data saving compared to total volume versus target $F1-score$

4.7 Conclusion

In the first part of this chapter we have studied the first attribute for the user behavior detection which deals with the user environment. As a first step, we have considered it as a binary classification problem, referred as IOD: Indoor Outdoor Detection. It is conducted from the network side (based on signals known by the Network). IOD task has been solved using a ML approach (based on Deep Learning) using 3GPP signals as features. So far, in the state of the art, works dealing with IOD using a ML approaches used $RSRP$ and $RSRQ$ as inputs. Whereas, the features that we used are: $RSRP$, CQI, TA, MI. We first showed the importance of this judicious choice of inputs. Replacing $RSRQ$ by CQI and adding both TA and MI to the $(RSRP, CQI)$ couple has shown an improvement of 10% in the overall performance of IOD. A diversified partially labeled dataset collected using a crowd-sourcing mode was used for evaluation. Such dataset allows to be as close as possible to the real behaviors of mobile users in daily life. Secondly, in order to also exploit the unlabeled data, we studied three semi-supervised machine learning approaches: Cluster-Then-Label, Co-Training, and Self-Training. A comparative study of these approaches was conducted. It showed that Self-Training (ST) approach is the best one for IOD. The ST training model obtained with a sharing of (19%,81%) between labeled and unlabeled data, and on a total volume of $250K$ of data, provides a $F1-score$ of 95%. We observe that if operators accept to reduce more than 1% the target $F1-score$ a saving of signaling load more than 30% is achievable. Such an evaluation - namely the required sharing between labeled and unlabeled data

for a target IOD performance - could be of interest to operators. Avoiding to tag all data strongly reduces the labeling efforts and constraints for the operators wanting to implement IOD algorithm. ST can thus perform well without requiring a complete labeling of data.

In the second part of this chapter, we showed that the mobile network can detect the user environment using a multi-class classifier trained on real data. This classifier uses standardised mobile network signals as input. We studied how to divide the initial indoor/outdoor classes further into more detailed environment types. However, training based on real data poses the problem that the data corresponding to different classes can be highly imbalanced. This imbalance is not due to the way with which we collected data collection, but is strongly related to the inherent user behavior and his preferences to services consumption. Knowing that this data imbalance can be a problem for machine learning, we tried to find a compromise between the data instances per class and the granularity degree by varying the number of environment classes. Thus, we studied different combinations of splitting the environment into different classes and compared their performance. The performance peaked for 2 class classifier as it is the easiest and for 4 class classifier which offers more detailed environment classification than just indoor and outdoor.

Results of the first attribute study answering the question "Where is the user while consuming a service" are very encouraging for further investigation. The next chapter will therefore deal with the second attribute related to the question "How is the user when experiencing a service? Static or Moving?".

Model	Optim.	layer	Drop.	Valid.	Epoch	Batch	Act.
IOD							
DL ($RSRP$, CQI, TA)	Nadam	8	NaN	30%	200	200	ReLu
DL ($RSRP$, CQI, TA, MI)	Nadam	8	NaN	10%	300	200	tanh
UED							
$2C$	RMSProp	7	0,4	20%	70	200	tanh
$3C_0$	RMSProp	7	0,2	20%	70	200	tanh
$3C_1$	Adam	7	0,3	20%	70	200	tanh
$4CO_0$	RMSProp	7	0,2	20%	70	200	tanh
$4CO_1$	Nadam	7	0,2	20%	70	200	tanh
$4CI_0$	RMSProp	7	0,2	20%	70	200	tanh
$4CI_1$	Adam	7	0,2	20%	70	200	tanh
$5C_0$	Nadam	7	0,2	20%	70	200	tanh
$5C_1$	RMSProp	7	0,2	20%	70	200	tanh

Table 4.9 – Summary of the models' hyperparameters: The optimizer, the number of hidden layers, the dropout ratio, the number of epoch and the batch size.

Chapter 5

Mobility Speed Profiles (MSP) Detection: How is the mobile user when experiencing a service? Static or Moving?

As a second step towards the unified AI-based model, for a joint detection of the two QoE-influencing user behavior (environment and mobility), we tackle the detection of second behavior attribute in this chapter. This attribute deals with the detection of mobility speed profile (MSP). In this study, we aim to answer the question "How is the user when experiencing a service? Is he static or dynamic?". The investigation of this behavior is studied progressively by going from detecting just 3 speed ranges, i.e., Low, Medium and High, to multi mobility speed ranges detection going to 8 speed ranges as described in figure 5.1.

In everyday life, average users are always moving and changing locations around three, four or much more times per day. Thus, there are different mobility speed profiles of mobile users. Analyzing and estimating them is of interest for the network. This helps in correct estimation of user speed giving important information linked to the consumption of network resources by user and his mobility. The cognition of speed will then help the network operator

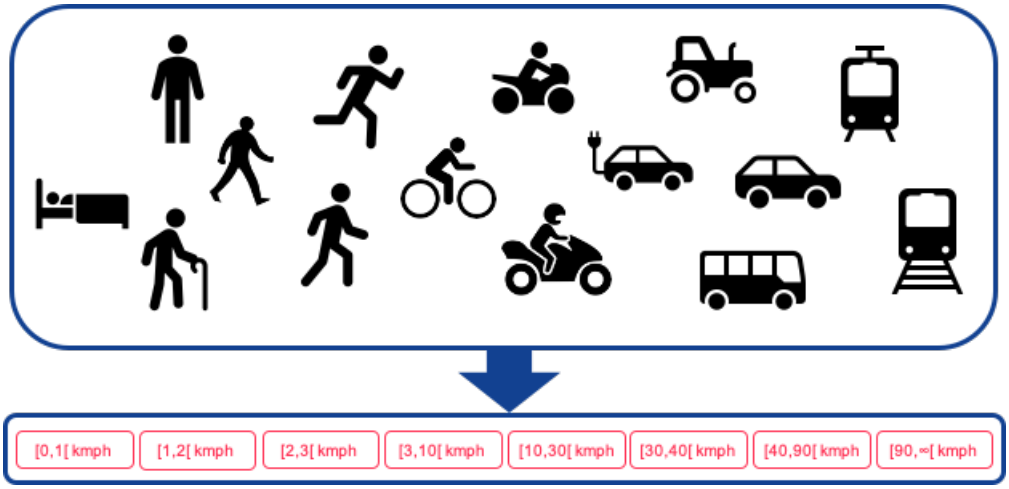

Figure 5.1 – Mobility speed ranges going to 8 speed ranges

to perform online network resource allocation and handover optimization according to users' speed profiles. Of course, it would be more precise if we are able to predict the user speed itself rather than detecting just his speed range. Nevertheless, from handover and resource optimization point of view, knowing the speed range instead of the speed itself can be sufficient. Thus, this chapter proposes Mobile Speed Profiling (MSP) investigation. We mean MSP to be speed profiling by detecting speed range of an active user, in real time. MSP model is learned using real data by classifying the user speeds to multiple mobility states, during connection. For MSP, we show that we can achieve a good performance even with a low and variable data collection frequency. Whereas, for speed regression, we will require more data for training, with higher and constant frequency, which will increase the network overload for collecting enough user-specific real data.

In literature, the issue of mobility state detection has been studied mainly considering 3 states: Low/ Medium/ High [95], [93]. However, mobile user speed profiling is more complex than considering only three states when targeting to infer the mobile user behavior. Actually, user's mobility conditions and thus the speed deeply affects the way he interacts with his mobile phone.

He usually tends to have different attitudes at home or at café than in transport, during the working week than during the week-end [34], etc. In chapter 1, we explained that this phenomenon is inherent to the mobile user behavior, which is about the preference to use their mobile phone in specific situations [35]. Therefore, user's behavior and experience in terms of service consumption varies according to his environment and mobility conditions. These two factors are thus important parts of user profile. Estimating the mobility profile in terms of only three states (Low, Medium, High) can be seen as a first level of analysis. However, it does not reveal the complexity of the situation. Thus, a more granular classification is desired in order to detect/predict at which speed profile a mobile user uses or prefers which mobile applications.

First, we tackle the MSP detection task for detecting 3-states mobility profiles, i.e, Low, Medium, High. Then, we expand our study by investigating the user mobility and speed profile detection using a supervised Deep Learning algorithm, based on a multi-output classification. This refers to the detection of multiple mobility or speed profiles using multi-class schemes. A class stands for a set of N (we target to have $N \geq 3$) successive speed categories bounded by minimal and maximal values. However the question is: to what level and detail the multi-class model requires to classify a user's speed? Can we classify the user's speed profile with more granular classes and with good performance? In the following, we provide a comparative analysis between various classification schemes of multiple classes. This will highlight the sets of classes with relevant labels. We will also show the trade-off between performance and more granularity.

This chapter is organized as follow: **Sec. 5.1** describes the main MSP works in literature. In **Sec. 5.2**, we study MSP as a 3 state profile which are the same profiles as proposed in literature and then we present some preliminary statistics. **Sec. 5.3**, analyzes user behavior and defines speed borders related

to his behavior. In **Sec. 5.4**, we propose the MSP schemes to study based on the preliminary analysis done in the previous section. In **Sec. 5.5**, we present the results of different MSP schemes. Finally conclusion is presented in **Sec. 5.6**.

5.1 State of the Art

In this section, we provide an overview of the prior research relevant to MSP detection in mobile networks. In 3GPP standards [95],[96], MSP is detected in a very simple way: i) Compute the number of handovers or cell re-selections during a sliding time window. ii) If this number is smaller than a threshold called "Medium", then the UE's mobility state is determined as "Normal". If this number is greater than "Medium" but less than a threshold called "High", the state is determined as "Medium". iii) Finally, if this number is greater than "High", then the state is determined as "High". The advantage of such MSP detection approach is that it is simple, easy to set up and less greedy in terms of resources and computing time. However, it is neither reliable nor accurate mainly with the massive deployment of pico and femto cells. Heterogeneous cell sizes complicate the relation between handovers and user speed.

In literature, many works have addressed the user mobility issue, mainly for handover management as well as for resource and energy optimisation reasons. To the best of our knowledge, only a very few works have studied user mobility from a context-awareness point of view. Some works like [93] and [94] have studied the mobility speed profiling by determining the speed category of a mobile user.In [93], authors propose to use $RSRP$ measurements to compute speed because signal fading over time is correlated with user speed. They propose two methods: one based on spectral analysis and other based on time-based spectrum spreading. For both methods, the signal variation is compared

to a reference curve or a look-up table (database) and a mapping is used to compute the UE speed. Authors in [94], propose a method for UE mobility estimation, relying on UE history of cell sojourn times. The neighbouring eNBs exchange among them the learned network topology as well as the history of UE sojourn times. Using such information, the eNB classifies the speed to one of the three mobility classes defined by 3GPP. Such solutions are interesting and show good performances, but they are rather complex and resource-intensive (checking a reference database each time or exchanging information between eNBs).

Other works in literature have also addressed the mobility issue, but they have rather proposed solutions for estimating the future position or the future cell of a user. They also compute the speed using this information. In [106], authors use the knowledge extracted from simple features such as cell ID and sojourn time in previously visited cells, to predict the user mobility and his future location. The prediction of the future position of the mobile user is computed using Machine Learning, more precisely using SVM which in turn is a classic machine learning tool. The work in [107] proposes a Markov chain based prediction technique to predict the next user position as well as his speed. Another work [108] uses a LSTM based system to learn the user mobility pattern from historical trajectories and to predict future movement trends of the user. According to the prediction results of the next user position, both [108] and [107] propose an algorithm to optimize the handover management. Another interesting work [109] evaluates the user mobility in 5G networks as a function of his behavior and his preferences. For evaluating mobility performance, they studied the user pause probability, user arrival, and departure probabilities.

According the QoE-influencing model of the user behavior (see chapter 1), guessing the user pattern of next cell, is out of scope of the 6 QoE-influencing attributes. However, solutions proposed in these works, especially the ones

using Machine Learning to detect mobility, are interesting in the context of our work. Besides, considering information like pause probability or user arrival or departure probabilities does not fit the assumption that we made in chapter 3: where the data features used for learning should be known to the network. Thus, the signals respecting our constraints are 1) $RSRP$ signal used by [93] and [94] and 2) the cell ID as well as 3) the sojourn time used by [106].

5.2 Data features

5.2.1 Description

We apply the same workflow described in chapter 3 on the MSP detection. Our requirement is to be able to detect MSP with a good accuracy and an F1-score greater than 95%. Next comes the second step of data collection. Thus, in our case for MSP study, the collected data comes from the same large crowd-sourced campaign (described in chapter 3). The data used for MSP is same as that used for IOD or UED ($RSRP$, CQI, MI, TA, Time), added to other features:

— ST: Sojourn Time in a cell.

— Extra Signals (feature engineered): signals derived from $RSRP$ and TA.

— GPS: Global positioning system measurements serve to automatically label the data used for Deep Learning training.

As a third step, we cleaned the data by replacing the null values by significant values and by eliminating the outliers. On the other hand, we also generate new features from TA and $RSRP$.

5.2.2 Data cleaning method for labeling

GPS measurements are collected at same time than the other data in order to automatically label the data used for Deep Learning training. In the case of

MSP detection, they contribute to build the ground truth or the labels needed for supervised learning.

Noisy labels: They determine the position of a user or device using the signals which are transmitted from up to 24 satellites. GPS works anywhere within the reaches of satellites, and it is much more robust than other location technologies. GPS positioning eventually found its way into the lives of millions (if not billions) of users with the development of Global Navigation Satellite System (GNSS) enabled car navigation devices and smartphones. The meter-level accuracy provided by GNSS receivers in smartphones enabled a wide range of location-based services including social networking, vehicle tracking, weather services and also for the purpose of many other applications like allowing the automatic detection of the critical aspects (mode of transport, purpose, etc.) of people's trips or like analyzing in real time the traffic networks, or the traffic speed and the huge number of traffic participants, etc [110]. One application field of GPS service and measurements that is of interest to us is: context awareness. Actually, such measurements are often used in the context-aware characterization since, they give a good information about the user location at a time t.

To collect this data coming from GPS measurements, the Android operating system allows us to access raw GNSS (Global Navigation Satellite System) measurements from smartphones or tablets through various APIs [111]. Making this data available opens up a world of possibilities to developers for the creation of new applications. However, in our case, we noticed that the collection via this API was very noisy and without continuous measurements. Some examples are presented in figure 5.2 (Red line). In fact, one of the biggest challenges for smartphone manufacturers is to increase battery life. Since continuous use of the smartphone's GNSS receiver would quickly drain the battery, the smartphone optimization process (developed by manufacturers), by brand,

employs a process known as duty cycling that aims to turn off the GNSS signals [110], [111]. As a consequence, it causes problem with reference since in phase of duty cycling GPS measurements done are at most of time false. Noise in GPS measurements will give rise to the issue of noisy labels. Recall that from Machine Learning point of view, label noise refers to examples that belong to one class that are assigned to another class. For example, in the figure 5.2, the case of a moving user at the left, is a pedestrian with a real speed not exceeding $3kmph$, but what has been erroneously detected with the collected GPS measurements is a speed between $15kmph$ and $30kmph$. This problem of noisy labels gets even worse, with dramatic effects, when facing imbalanced classes. Given that, examples in the few populated classes are so few, losing some to noise reduces the amount of information available about the minority class. Additionally, having examples from the majority class incorrectly marked as belonging to the minority class can cause a disjoint or fragmentation of the minority class which was already sparse because of the lack of observations. Hence the need is to correct these measurement errors as much as possible. As we use real data, the GPS measurements are noisy, namely some data are erroneous. For this reason, GPS data should be cleaned in order to ignore measurements that we detect as erroneous.

Label computation: The user speed, for labeling, is calculated based on these measurements and time values. In order to automatically label, a typical approach is to transform the series of measurements that record position points (latitude and longitude) at regular time intervals in coordinates (x, y) in km. To derive the average mobile user speed v, we use a succession of N coordinates (x, y) derived from GPS information. Note that this is only done during data labeling phase and GPS information is not assumed to be available during classification phase. Imagine that the mobile user moves along a path P from point A to point B (see figure 5.3). To link these two extreme points, the

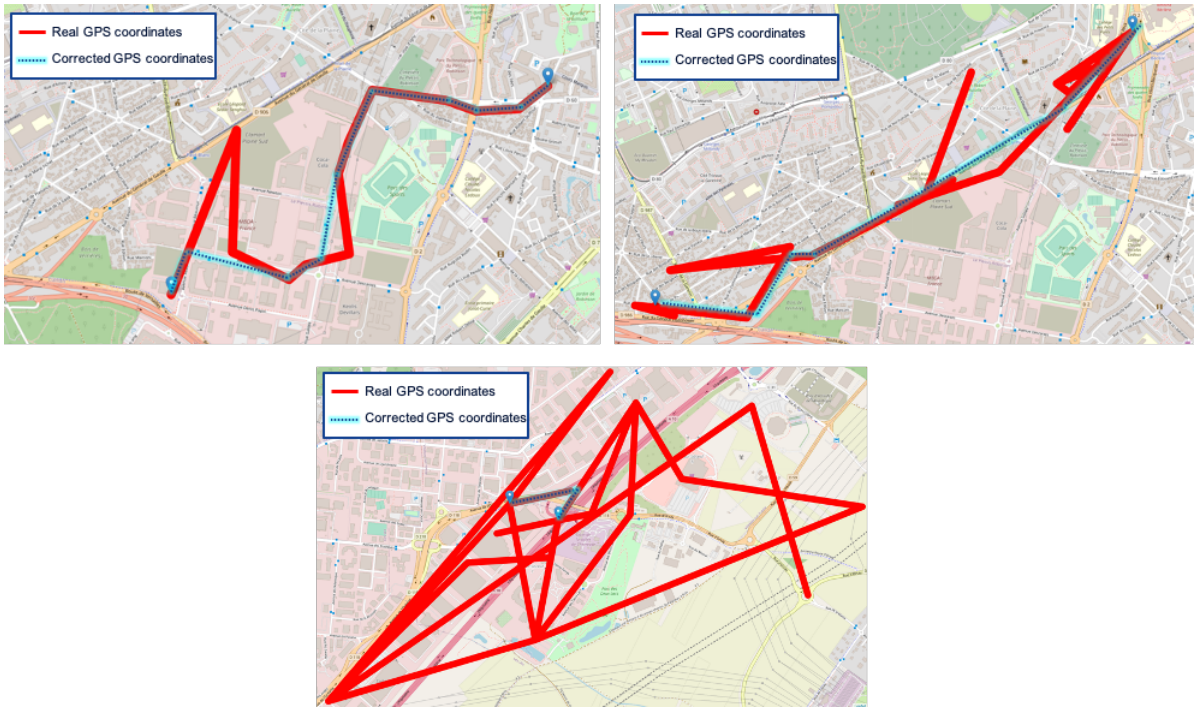

Figure 5.2 – Examples of the collected GPS measurement. Red line depicts the real collected data (very noisy). Blue line depicts the corrected trajectory using our proposed Algorithm.

user goes through $N-2$ intermediate points belonging to $\mathrm{S} = \{a_i\}_{i \leq N-1}$ at time $\{t_2\}_{i \leq N-1}$.

Let TCR_{max} be the elapsed time to go from A to B and L the total distance. Let δt_i the elapsed time and l_i the distance between $\{a_{i-1}, a_i\}$. The point a_i has the following coordinates $(x(t_i), y(t_i))$. $l_i = \sqrt{(x(t_i) - x(t_{i-1}))^2 + (y(t_i) - y(t_{i-1}))^2}$ and $\delta t_i = t_i - t_{i-1}$.

The average speed $\hat{v}$, is approximated as:

$$\hat{v} = \frac{1}{N-1} \sum_{i=1}^{N} \frac{l_i}{\delta t_i} = \frac{1}{N-1} \sum_{i=1}^{N} \hat{v}_i$$

$$\text{with } t_N - t_0 = TCR_{max}$$

A step of labeled data cleaning is primordial since it ensures the integrity of

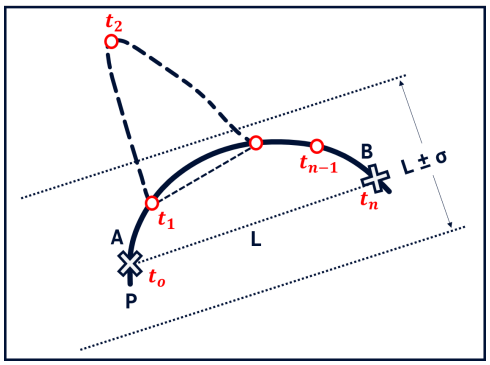

Figure 5.3 – Speed Computing between according to path P between two points A and B during a sliding window TCR_{max}

the ground truth used for labeling. For this, let σ be the threshold that sets a confidence interval. At t_i, if $l_i \in [L \pm \sigma]$ then $a_i \in \mathrm{S}$ otherwise $a_i \notin \mathrm{S}$. The distance l_i between 2 successive points should be bounded by L to be used for the average speed derivation. Otherwise the measurement at time t_i is considered as an outlier and is excluded. The computing of labels is detailed in Algo. 1.

Algorithm 1 LABEL COMPUTING

$(x, y) \leftarrow$ transform(lat,long,TCR_{max})
$L_{AB} \leftarrow \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$
Initialize σ, $Index \leftarrow 0$ And $v \leftarrow 0$
for i in $[0, TCR_{max} - 1]$) **do**
    $l_i \leftarrow \sqrt{(x_{i+1} - x_{Index})^2 + (y_{i+1} - y_{Index})^2}$
    if l_i in $[L_{AB} \pm \sigma]$ **then**
        $v \leftarrow v + \frac{l_i}{t_{i+1} - t_{Index}}$ And $Index \leftarrow Index + 1$
    else
        Consider Measure i as an outlier and skip it
    end if
end for
$V_{TCR_{max}} \longleftarrow \frac{v}{Index+1}$
Label Coding according to $v_{TCR_{max}}$

Once the speed is computed, the instance can be associated to the right label. Now, the question is: what is the most appropriate value of TCR_{max} (width of the sliding window) to compute the speed? Considering micro-cells, mostly

represented in urban environments, we assume typical cell radius distances between 0.2km and 2km. For both values, curves of mobile user speed versus cell crossing time is plotted in Fig. 5.4. They enable us to fix TCR_{max} at 10 kmph where there is a clear separation between CDFs of MI. So, the duration TCR_{max} for calculating the speed is empirically set to 300 seconds.

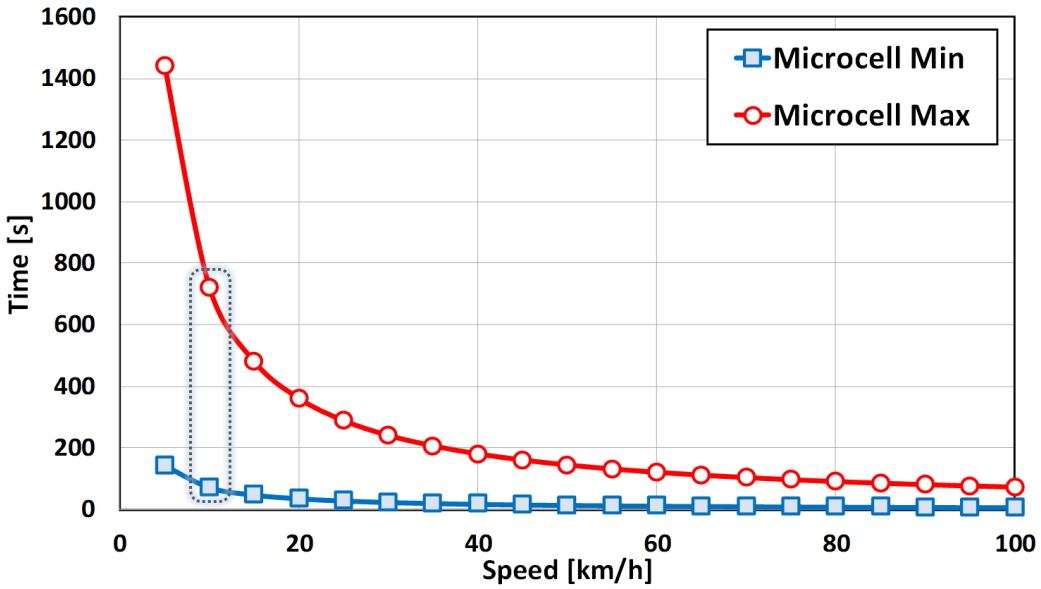

Figure 5.4 – Speed versus time for two cell radius

5.2.3 Data augmentation method for balancing the dataset

We observe in figure 5.6 of section that higher number of classes reduces the number of instances of data in each class which leads to a problem of imbalanced classes. Imbalanced classes can pose particularly a problem during the training phase of any ML algorithm and not only for neural networks [66]. The challenge of working with imbalanced datasets is that most machine learning techniques will ignore, and in turn have poor performance on the minority class. Although typically it is performance on the minority class that is most important. One approach to address imbalanced datasets is to over-sample the

minority class. The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples. Creating or oversampling data in minority classes is called data augmentation. Data augmentation is a technique to artificially create new training data from existing training data. This is done by applying domain-specific techniques to examples from the training data that create new and different training examples.

To resolve the imbalanced class issue, we focus on the Artificial Data Augmentation (A-DA). A-DA learns the distribution of a stratified sample from our dataset and then generates data with similar statistical properties. This type of data augmentation is used for the minority class and is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short. SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line [112], [113].

5.3 Preliminary analysis: user activity vs speed category

5.3.1 Speed category definition

A speed category is defined by a minimal and maximal speed value. Speed categories are highly influenced by many factors including the environment type as well as the time of day, which complicates the detection task. The boundaries of categories are extracted from the following set denoted as B:

— $\mathrm{B} = \{0, 1, 2, 3, 10, 30, 40, 90, \infty\}$ (in kmph).

They were selected in order to reflect the complexity of a user's daily life and

capture the variety of his movements in real world. They represent the typical speeds given in [114], corresponding to various environments met by mobile users (urban, rural, highways, road, pedestrian, bus, car and train).

The figure 5.5 shows the environment type distribution versus the speed category. Note that the borders shown are chosen from $\mathbb{B}$. We observe that some boundaries are relatively much more visible between some speed categories according to the type of environment. We notice such prominent boundaries at 10 kmph and 40 kmph.

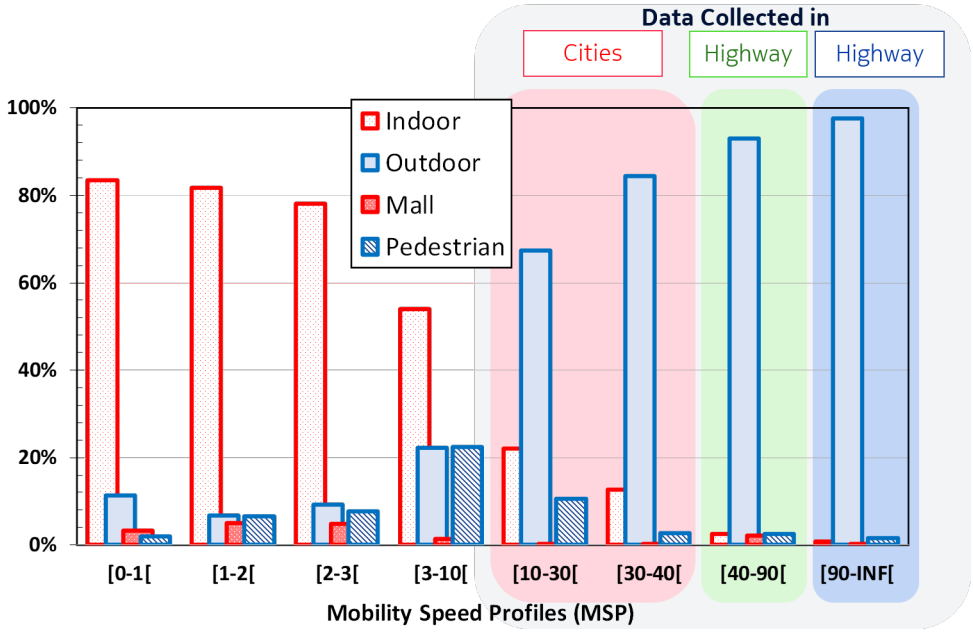

Figure 5.5 – Environment type distribution vs speed category (real data)

The 10 kmph boundary makes the split between indoor and outdoor that are two distinct environment types with different physical characteristics. The points below this boundary represent more than 60% (up to 80%) of total points. They belong to $[0,1[$ kmph, $[1,2[$ kmph, $[2,3[$ kmph speed profiles measurement points collected inside buildings. Whereas, the instances above 10 kmph have been mostly collected outdoor: between 10 and 40 kmph in cities and above 40 kmph on highways. We also notice in figure 5.5 that there are some errors

as this is real data. For example, we note some point of measurements where the user is really indoor, but the speed is higher than 40 kmph which is logically impossible. This is addressed by data cleaning algorithms for the labeling process.

5.3.2 Relation between user activity and speed profile

Figure 5.6 depicts the user activity by plotting the phone usage ratio for different speed categories derived from the boundaries in $\mathbb{B}$. Note that a user's activity is characterized by his phone usage. The 'phone usage' state is defined as the state when the user uses his phone or equivalently the screen is on as well as unlocked and there is data exchanged. User activity per category is then measured as the ratio between the number of instances per category, when the user has been using his phone effectively, and the total number of activity instances. In fact, figure 5.6 illustrates the percentage of total time the user is connected to 4G network and is exchanging data. We observe that most of the user activity occurs when the user moves at a speed lower than 1 kmph (75% of activity), which is mainly when the user is indoor or is walking as a pedestrian.

Figure 5.6 highlights the user activity trends that we observed after statistical analysis on mobile user behavior in literature.

During daytime, a user experiences different situations, such as walking outdoor, in a car, at work, in a mall, in transport or at home. Actually, mobile users' preferences for certain applications or contents are linked with his current usage situation [34]. In literature, some statistical studies show that mobile phones are mostly used for internet service (80% of data calls) as well as for calls (70% of voice calls) [35] when a user is inside a building. This can be explained by the fact that different use contexts pose their own limitations and, thus, impact the potential application usages. In [28], [36], the most com-

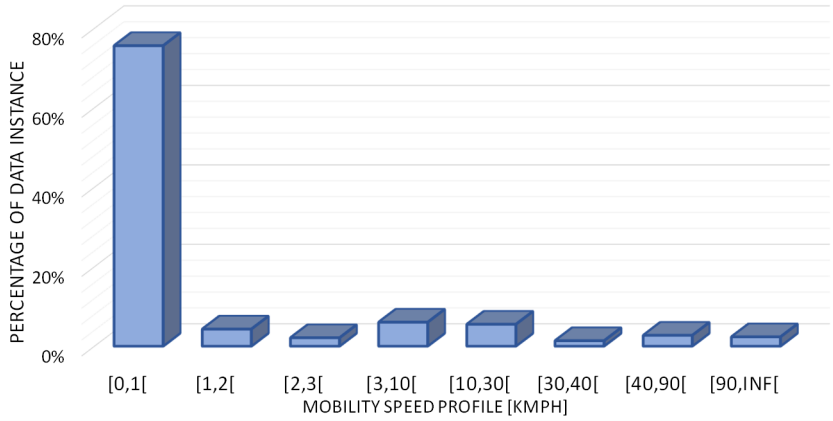

Figure 5.6 – User activity per mobility speed profile

monly mentioned physical environments, for application usage, are indoor, but they also include vehicles, such as public transportation and private cars. The home environment is mostly preferred by users. Moreover, as soon as the user speed increases, the network quality deteriorates and consequently the user may switch from 4G to 3G or even to 2G (rarely). Consequently, there exists more data in the lowest speed category as compared to other categories, in terms of phone usage. We also observed in Fig. 5.6 that the data instances are distributed unequally between different categories. Thus, designing a speed classification scheme can face the problem of unequal classes in terms of data distribution.

5.3.3 User activity during daytime per speed category

We analyze the ranges of mobility at various time slots in a day: during the working days, weekends and finally in different periods of day (Morning, Afternoon, Evening, Night). Figure 5.7 shows the evolution of user activity by plotting the phone usage ratio versus the hour of day for various speed categories. We notice again that most of the activity is done more at lowest speed, but at certain time slots of the day. Actually, the lowest speed corresponds to

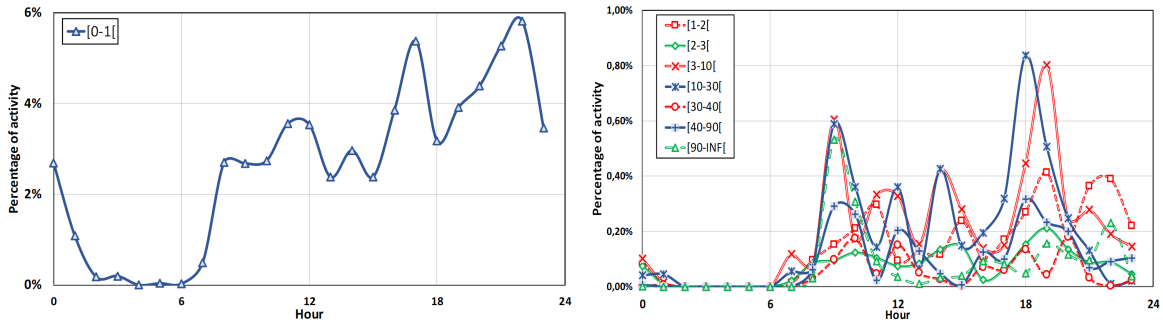

Figure 5.7 – User activity vs. hour per mobility speed profile

the static or almost-static state of a user. This is the user's preferred situation to request mobile services. We also observe a certain reproducibility of the phone usage for all categories during a day. Effectively, during the night most of people are static. They start to be active when moving during the day. The activity is very high for the lowest speeds between $8a.m.$ and $11p.m$ and is punctually high at certain time slots that corresponds particularly to the transport instant between work and home or other locations.

To sum up, so far, preliminary analysis, of the user activity vs. the MSP over time, and across different environment, have pointed out three points to be investigated in order to make the MSP detection more accurate. These points are as follow:

— Border Investigation: 40 kmph and 90 kmph are not necessarily the best borders of the user speed profiles. We have noted for example, the importance of the 10 kmph as border since it's kind of a separation between the two environments: indoor and outdoor.

— Noise in labels: Label noise refers to examples that belong to one class, but are erroneously assigned to another class, which are many in our case. Two types of noise are distinguished in the literature: feature (or attribute) and class noise. Class noise is generally assumed to be more harmful than attribute noise in ML [67]. Actually, it makes determining

the class boundary in feature space problematic for most machine learning algorithms, and this difficulty typically increases in proportion to the percentage of noise in the labels.

— Imbalanced dataset: Until today, this imbalanced nature of classes still remains a challenge for ML algorithms. In general, the more the imbalance is important the more the classification result will be dramatic. For the UED detection task, we have also noted the imbalanced dataset issue. However, it is much more present in the MSP. In our case, our dataset has very skewed class distribution with 80% of instances in one class and the remaining 20% distributed, roughly equally, in the other 7 classes. As a matter of fact, this is not related to the data collection constraints, but it is rather, inherently, due to the nature of real human activities. As a consequence, in order to ensure an efficient scheme of classification for real user activity, we have to find a trade-off to limit this inherent data bias for different classes.

In the rest of this chapter, we will investigate these points and evaluate MSP performance in order to reach our objective of MSP detection with an F1-score of around 95%.

5.4 Classification schemes

The objective of this section is to define the relevant classification schemes by smartly regrouping various speed categories chosen from the set B. In order to ensure an efficient scheme of classification for real user activity, we have to find a trade-off to limit this inherent data bias for different classes. We aim to design a classification scheme that detects detailed speed profiles of mobile users, with a fine granularity and a low decision error margin.

Problematic 6

What are the best speed profile borders for the MSP detection task?

Contribution 6

We propose to fix the speed profile borders that impact the user activity most while ensuring a trade-off between the number of classes and the imbalance issue.

We then consider different possibilities, ranging from a simple 3-state classification problem to a more detailed classification of a user's speed. For that, we decide to regroup the speed categories based on the similarity between the cumulative distribution curves of the collected data $\{RSRP, MI\}$. These 2 features were identified to be contributing the most after we ranked them using Principal Component Analysis method. Thus, we based our analysis mainly on these features.

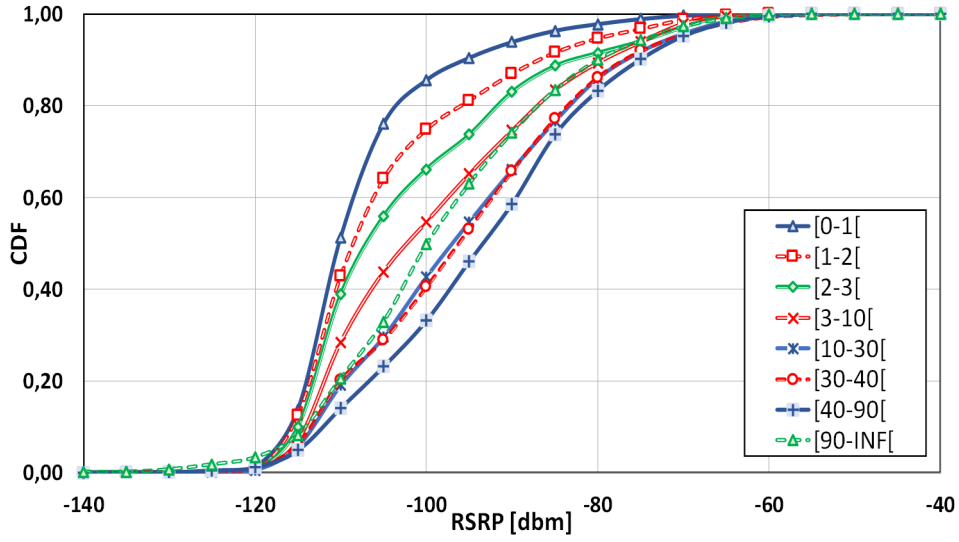

Figure 5.8 – Empirical CDF for measured RSRP per speed category

The created merged groups shall also include data with similar statistical

properties in order to optimize the classification results. Figures 5.8 and 5.9 depict the cumulative distribution curves vs. $RSRP$ and MI, respectively. We observed from these curves that 4 groups of similar curves can be extracted: a set with only $[0,1[$ kmph, a set regrouping $\{[1,2[$ and $[2,3[\}$ kmph, a set alone with $[3,10[$ kmph, and another assembling $\{[10,30[$, $[30,40[$, $[40,90[$ and $[90,\infty[\}$ kmph. This results into a clear split in 2 groups: one associated to users moving very slowly (walking or static) and the others moving at high speed. This separation is more noticeable in the CDF of MI.

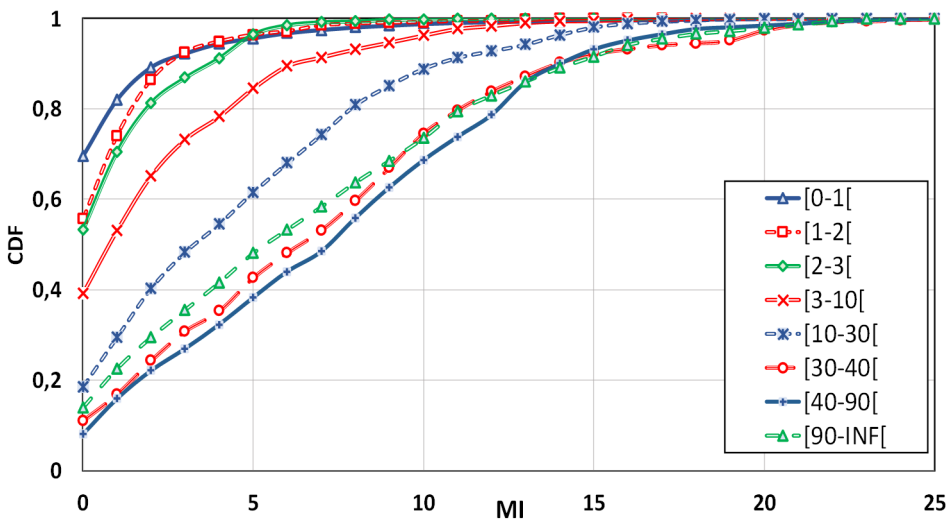

Figure 5.9 – Empirical CDF for measured MI per speed category

Based on these observations, we propose to investigate ten schemes of 3, 4, 5 and 8 classes (3C, 4C, 5C and 8C). A class corresponds to a set of speed categories. Thus a class is defined by minimal and maximal boundaries of the associated speed categories. Let a multi-class scheme i be composed by a set of classes denoted as $\mathcal{C}^i = \{C_1^i, C_2^i, ...C_n^i...\}_{n \leq N_i}$ with N_i the number of classes.

Let $\mathcal{B}_i$ the set of boundaries coming from the speed categories selected. If,

$$\mathcal{B}_i = \{B_1^i, B_2^i, ...B_j^i...\}_{j \leq N_i+1}$$

with B_j the speed value in kmph. Then, the class n of the scheme i is written as:

$$C_n^i = [B_n^i, B_{n+1}^i[\ \forall \ n \leq N_i$$

with C_n corresponding to speed values in the interval $[B_n, B_{n+1}[$.

A plurality of speed categories, containing different combinations, are then investigated to better model the diversity of speed situations, thus defined by the optimal boundaries of the categories:

Scheme	Boundaries kmph	Speed range kmph
$3C$	$\{0, 10, 90\}$	$[0, 10[$, $[10, 90[$, $[90, \infty[$
	$\{0, 40, 90\}$	$[0, 40[$, $[40, 90[$, $[90, \infty[$
	$\{0, 3, 90\}$	$[0, 10[$, $[10, 90[$, $[90, \infty[$
	$\{0, 3, 30\}$	$[0, 3[$, $[3, 30[$, $[30, \infty[$
$4C$	$\{0, 1, 10, 90\}$	$[0, 1[$, $[1, 10[$, $[10, 90[$, $[90, \infty[$
	$\{0, 1, 3, 30\}$	$[0, 1[$, $[1, 3[$, $[3, 30[$, $[30, \infty[$
	$\{0, 1, 3, 90\}$	$[0, 1[$, $[1, 3[$, $[3, 90[$, $[90, \infty[$
$5C$	$\{0, 1, 3, 10, 90\}$	$[0, 1[$, $[1, 3[$, $[3, 10[$, $[10, 90[$, $[90, \infty[$
	$\{0, 1, 10, 40, 90\}$	$[0, 1[$, $[1, 10[$, $[10, 40[$, $[40, 90[$, $[90, \infty[$
$8C$	$\{0, 1, 2, 3, 10$	$[0, 1[$, $[1, 2[$, $[2, 3[$, $[3, 10[$
	$30, 40, 90\}$	$[10, 30[$, $[30, 40[$, $[40, 90[$, $[90, \infty[$

Table 5.1 – Mobility Speed Profiling - MSP classification schemes

5.5 Supervised Deep Learning-based classification performances

5.5.1 Architecture and configuration

For our supervised multi-output classification problem, we propose to investigate DL-based model for MSP. The FNN model is built with around 300k

instances of LTE data collected only in France. Note that our data size increased over time. It has been trained using 70% and tested on the remaining 30%.

The system architecture is shown in figure 5.10. It is a Feed Forward Neuronal Network (FNN) composed of 3 main parts:

— Input: A first input layer is fed with 10-features described in section 5.2 that are 4G collected data.

— Core: Containing 5 hidden layers as well as a dropout layer to regularize and minimize the over-fitting.

— Output: An output layer with either 3, 4, 5 or 8 classes (Depending on the classification scheme).

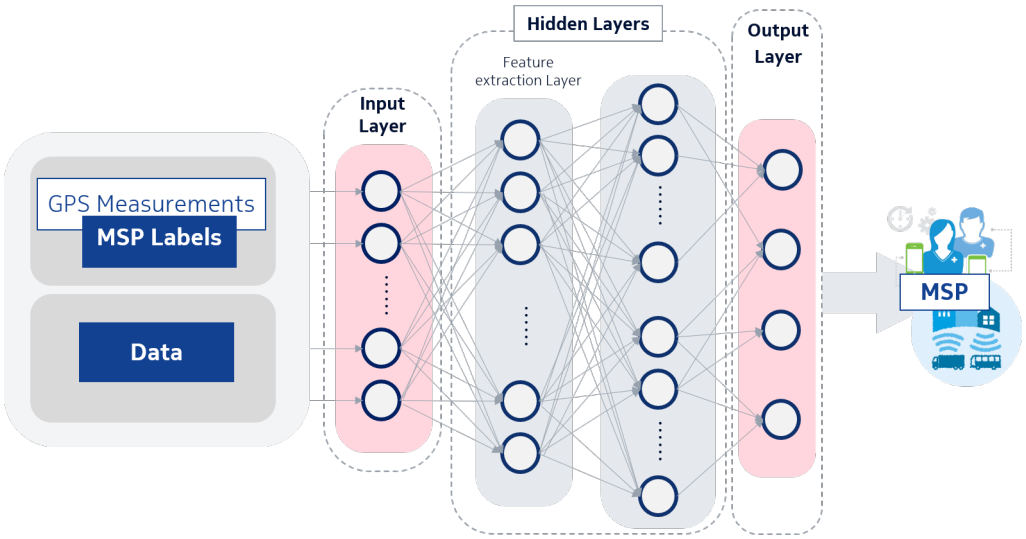

Figure 5.10 – Mobility Speed Profiling (MSP) system architecture - Note that there are several hidden layers and not just two as it may initially appear.

The MSP is implemented with both Scikitlearn and Keras, using Tensorflow as the back-end engine, for the computation and development of the DL models.

Recall that, a crucial step of any neural networks implementation is to optimize the hyperparameters set for the model, i.e., the number of hidden layers, batch size, epoch size, the weights, the activation function, the loss function, the learning rate etc. The challenge with these hyperparameters is that there

Optim.	layers	Dropout	Valid.	Epoch	Batch	Act.
Nadam	5	0,1	30%	200	80	tanh

Table 5.2 – MSP hyperparameter: The optimizer, the number of hidden layers, the dropout ratio, the number of epoch and the batch size.

is no magic combination that always works. The best combination of hyperparameters depends on each task and also on each data set. Recently, a new approach using Bayesian optimization for tuning hyperparameters has been considered (see chapter 3). As compared to other approaches, the Bayesian method has the property to rapidly reach the optimal set of hyperparameters. For this study, the hyperparameters of the DL solution have been tuned using the Bayesian optimization (see Appendix A for hyperparameter methods comparison).

As proved experimentally in appendix B, tuning the hyperparameters for one scheme and use the set of the resulting hyperparameters for the other schemes (instead of running an optimization per scheme) allows to save computational time. The performance losses between these 2 scenarios are minimal (around 0.2% of decrease in F1-score) compared to the gain in terms of calculation time. Thus, in this chapter we will optimize the hyperparameters for one classification scheme. And the resulting hyperparameters set will be used for other classification schemes. The hyperparameter set that has been used is described in the table 5.2. The hyperparameter optimization was conducted on the classification scheme $\{0, 10, 90\}$. We set out success-criteria to 95% of F1-score.

5.5.2 Results

Table 5.3 presents the performance results of the multi-output classification schemes given in 5.1 and with hyperparameters given table 5.2.

As a first step, we created the mobility tags from GPS coordinates to create 3 labels: low, medium and high like in [94] and [93]. The low label corresponds to a speed range between $[0, 40[$ kmph, the medium and the high labels correspond respectively to the speed ranges $[40, 90[$ kmph and $[90, \infty[$ kmph. No data label cleaning has been applied. For training, we used the same data volume as used in chapter 4, in section 4.2, which is equal to $72k$ measurement points. Alternating between step 4 of learning and step 5 of optimization (see ML workflow in chapter 3), the best model has delivered an F1-score of 78.30% [43]. Such a score, is far away from our success criteria fixed to 95% of F1-score hence the need for a retro-back action which brings us back to step 3 with the following questions:

— Are the borders 40 kmph and 90 kmph limiting the 3 profiles low, medium and high the right ones?
— Should we only consider 3 speed profiles?
— What borders best reflect the user activity?
— Is the labeling step by transforming the GPS coordinates into a speed profile been done properly?

As a matter of fact, a better MSP detection can be achieved when analysing the user's activity according to the speed profile and the user environment over time. Clearly, analysing our data gives us a clear picture of a user's activity in different profiles of speed. Thus, we can fix the suitable mobility speed profiles' borders in order to optimize the detection and reach our success criteria. Furthermore the performance can be improved when label data is cleaned.

Simulations are done for all the MSP classification schemes. The results shown in Table 5.3 presents the accuracy and F1-score metrics for MSP. First, in order to quantify the added value of correcting labels, we compare the results of MSP based 3 profiles obtained with no label cleaning, to the same classifi-

	3 Classes							
	$[0, 10, 90]$		$[0, 40, 90]$		$[0, 3, 90]$		$[0, 3, 30]$	
	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.
FNN	**98.367**%	**94.256**%	98.222%	93.861%	97.226%	93.570%	96.956%	93.038%
FNN + A-DA	**98.096**%	**96.012**%						
	4 Classes							
	$[0, 1, 10, 90]$		$[0, 1, 3, 30]$		$[0, 1, 3, 90]$			
	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.		
FNN	**95.365**%	**91.167**%	95.004%	89.917%	95.517%	90.447%		
FNN + A-DA	**95.968**%	**93.559**%						
	5 Classes				**8 Classes**			
	$[0, 1, 2, 10, 90]$		$[0, 1, 10, 40, 90]$		$[0, 1, 2, 3, 10, 30, 40, 90]$			
	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.		
FNN	94.484%	87.285%	**94.913**%	**89.066**%	**93.321**%	**81.108**%		
FNN + A-DA			**94.283**%	**92.092**%	**93.933**%	**85.484**%		

Table 5.3 – Deep Learning-based supervised multi-output classification performance using a Feed Forward Neuronal Network (FNN) with and without Artificial Data Augmentation (A-DA): F1-score vs. classification schemes

cation scheme after label cleaning. We compare both schemes of $\{0, 40, 90\}$ with and without correcting labels on the scheme. Results show a gain of around 15% (from 78.3% to 93.8% of F1-score), while using the label correction algorithm 1. Indeed, the better the learning data is (the same for input and labels), the better the performance is.

We observe that we have an overall good performance. All the considered classification schemes show an accuracy higher than 93% and an average F1-score of 90%. However, as we expected, we also observe that the higher the number of classes is, the lower the performance is. This can be explained as

follows: increasing the number of classes reduces the number of instances of data in each class which worsens the problem of imbalanced classes. The effect of imbalanced classes increases if the boundaries between classes are very similar, which is the case for us.

To resolve the imbalanced class issue for the MSP task, we employed the Artificial Data Augmentation (A-DA) technique, which is referred to as the Synthetic Minority Oversampling Technique, or SMOTE for short. SMOTE works by selecting examples that are near in the feature space. It draws a line between the examples in the feature space and creates a new sample at a point along that line. To be more specific, first a random example from the minority class is chosen. Then k of the nearest neighbors for that example are found (typically $k = 5$). A randomly selected neighbor is then chosen and a synthetic example is created at a randomly selected point between the two examples in feature space [112]. For our use of A-DA, we limited the generation of the artificial data to 65k in order to stay close to the original data and to avoid additional noise. The impact of artificial data augmentation (A-DA) on the best 3 classification schemes, with FNN (cases in bold), shows an improvement of around 3.0% in terms of F1-score. This is explained by the fact that FNN is greedy: more data results in better performance, as argued in [31] and [32].

We observe that certain boundaries are common in schemes with higher MSP performance. The best schemes per class (in bold) are those which fix the boundaries to 1, 10, 40 and $90kmph$. Indeed, the $10kmph$ boundary splits the points between indoor and outdoor. Therefore, the classes on both sides of the $10kmph$ boundary are well detected because they are differentiated by the environment of the location where the data was collected. Furthermore, $[10, 30[$, $[30, 90[$ and $[90, \infty[$ present various high speed profile environments (cities and highways, in urban as well sub-urban environments). Moreover, introducing a $1kmph$ boundary brings enhancement. Indeed, instances in the $[0, 1[$ class

are very well detected since this class is the most populated class. But, the smallest classes $[1, 2[$, $[2, 3[$, $[3, 10[$ are detected with lesser accuracy since they mostly present similar physical properties and are confused with $[0, 1[$. Thus, we show using real data that a granular learning of the user mobility speed profiles can be achieved with good performance.

5.6 Conclusion

In this chapter, we have investigated the mobility speed profiling (MSP) of the user while consuming a service. We have proposed an intelligent system for MSP detection linked with user activity preference. We used a FNN network to learn the user mobility profile when a given user is active and using his phone. Based on crowd-sourced user-specific data, we achieved above 94.5% of detection in terms of F1-score in existing deployed 4G Heterogeneous Networks (HetNets). We mean by HetNet the diverse and multiple types of cells: macro-cells, picocells, femtocells, etc. We worked with real data using GPS coordinates as labels. They must be cleaned because of inherent noise. This noise can impact the overall performance of any ML algorithm. We observed that the speed profiling is influenced by the user environment. This is because signals used for MSP are impacted by the environment of the data collection location. Thus, for learning more details, it is necessary to pay attention to the environment type in order to fix the boundaries for the MSP task. Furthermore, we noted that the level of imbalance in data influences the F1-score. We found that one way to overcome this issue is to collect more data or slightly re-balance the data artificially.

Chapter 6

Multi-Task Learning for Joint Detection of User Environment & Mobility

So far in this thesis, we have investigated both the environment and the mobility speed range of mobile users while they are consuming services. Actually, both user's environment and mobility situation are important factors since they have a big influence on QoE. For example, a user who is indoor would experience a very different service quality as compared to users who are outdoor, all else being equal. The environment and the mobility make the conditions in which a mobile user consumes the requested services/applications. In practice, in order to estimate the environment or mobility, we have to answer the following questions: how and where a mobile user consumes the mobile services? Also, these questions have to be answered at the same time (simultaneously) to detect and then predict the user behavior. Thus, we investigate a joint detection scheme by association of both user environment detection (binary/ multi-environment detection) and the mobility speed profiling detection (MSP).

One way to detect these two attributes simultaneously is to use the transfer learning approach and more precisely the Multi-Task learning (MTL) which is a subcategory of transfer learning. Using MTL allows to solve multiple learning

tasks at the same time, while exploiting commonalities and differences across tasks. Such approach, is the best solution for our problem because it allows to answer multiple questions simultaneously. Besides, learning multiple tasks jointly improves the generalization on other tasks and brings a mutual benefit [115].

In this chapter, we investigate a Multi-Task Deep Learning solution to jointly detect a user's environment and mobility profile. The empirical evaluation based on real-time and highly representative radio data shows the effectiveness of our approach. This data includes ground truth information and the whole dataset has been massively gathered from many diverse mobility situations and many environment types. The results also prove that the simultaneous detection of the environment as well as the mobility state estimation can be achieved with high accuracy.

This chapter is organized as follow: **Sec. 6.1** presents the basics of Multi-Task learning approach: beginning from the definition, to the motivation and coming to the Multi-Task existing types. In **Sec. 6.2**, we present our adopted architecture to jointly solve the user environment and the user mobility range while consuming a service. **Sec. 6.3** enumerates the inputs used for the MTL. In **Sec. 6.4**, we present and discuss the performance of the MTL approach to jointly detect the two users attributes. Finally, conclusion is presented in **Sec. 6.5**.

6.1 What's Multi-Task Learning?

6.1.1 Definition

Generally, people have an inherent ability to transfer knowledge across tasks. What they learn or acquire for solving one task, they use it in the same way to solve one or many other tasks that are relatively linked directly or indirectly

with the first acquired one. The more related the tasks, the easier it is for us to transfer, or cross-utilize our knowledge. For example, we use the algorithmic and the theoretic knowledge, which serves as a base acquired at school. It server us further to learn different programming languages. As humans, we often use the same basics that provide us with the necessary skills to learn several tasks and master more complex techniques. In other words, we don't learn everything from scratch when we attempt to learn new aspects or topics. We transfer and leverage our knowledge from what we have learnt in the past.

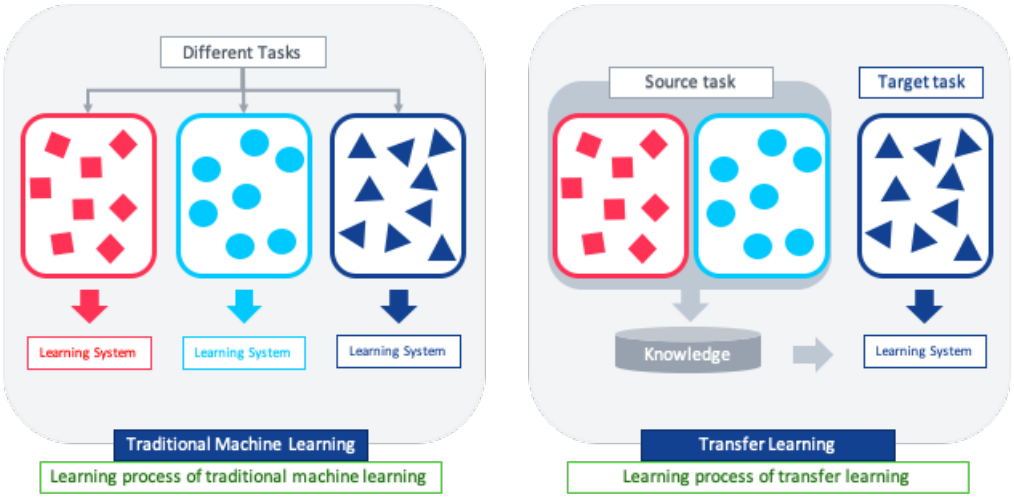

Figure 6.1 – Different Learning Processes between Traditional Machine Learning and Transfer Learning according to [116]

Conventional machine learning and Deep Learning algorithms, so far, have been traditionally designed to work in isolation. These algorithms are trained to solve specific tasks. The models have to be rebuilt from scratch once the feature-space distribution changes. Inspired from the human nature and abilities, Transfer Learning (TL) is an idea of overcoming isolated learning systems and use knowledge acquired for one task to solve the other related ones [116]. In fact, TL paradigm has aroused the curiosity of scientists for a while: its first appearance was in 1995 in the Neural Information Processing Systems (NIPS) workshop [117]. With DL emergence, TL again became an active research issue

around the 2010s. It has been considered in literature as a means of transferring knowledge from a source domain to a target domain (see figure 6.1. Unlike traditional machine learning and semi-supervised algorithms, transfer learning considers that the domains of the training data and the test data may be different [118].

Definition 2: Transfer Learning

Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T()$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.
Where:

- A domain D consists of two components: a feature space X and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in X$.

- Given a specific domain, $D = \{X, P(X)\}$, a task consists of two components: a label space Y and an objective predictive function $f()$ (denoted by $T = \{Y, f()\}$), which is not observed but can be learned from the training data, which consist of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y$. The function $f()$ can be used to predict the corresponding label, $f(x)$, of a new instance x. From a probabilistic viewpoint, $f(x)$ can be written as $P(y|x)$.

According to the above definition, transfer learning techniques can be divided into many categories (for further information see references [116] and [118]). Let's assume in our case that the source domain is the environment study and the target domain is the mobility study. We consider the environment as the source domain since the environment labels are more accurate than the mobility labels. Thus, in our case, $D_S = D_T$ and $T_S \neq T_T (Y_S \neq Y_T)$ and both Y_S and Y_T are available which leads us to the Multi-Task (MLT) subcategory of TL.

In other words, MLT learning aims to transfer knowledge between two tasks while learning both of the source and target tasks jointly and efficiently.

6.1.2 Why Multi-Task works?

Learning tasks jointly is very beneficial to get a final model that will be capable of exploiting the relatedness of the tasks to improve its generalisation accuracy. This is can be explained by:

1. Statistical data augmentation/ Implicit data augmentation [119]: In general, the used data for any task is always noisy (the noise level differs from one dataset to another and from one task to another). While applying a machine learning approach on a given dataset, we are opting to learn the best ML model that learns a good representation per task ignoring the data-dependent noise and generalizing well. We mean by generalizing well that the learnt ML Model has a good abilities to adapt properly facing new data that was not seen before (during the training). Knowing that the noise pattern is different from one task to another, so learning multiple tasks simultaneously helps to increase the ML model generalization capacity. Indeed, the joint learning average the noise between the different learnt tasks. Thus, in return, reduces the over-fitting risk.
2. Attention focusing [119]: For ML approaches, generalization from a high dimensional feature space and low dimensional dataset is very hard since the data volume is not enough to properly learn how to map the output space to the input space. Usually, facing this issue, some dimensionality reduction approaches are studied to compact or reduce the feature space losing by that some information that can help increasing the ML performance. However, learning Multi-Task jointly can be beneficial to this issue. Actually, the knowledge brought from other tasks can help the model

to differentiate between relevant and irrelevant features. Thus, MTL can help the model to focus its attention on those features that actually matter as other tasks will provide additional evidence for the relevance or irrelevance of those features.

3. Eavesdropping [120]: Let us assume that there is one feature from the features set that is useful for two tasks. This feature is learnt easily from the first task, but the second task uses it in a more complex fashion. In that case, the second task will eavesdrop on the internal representation from the first task to learn that feature better.
4. Feature selection double check [115]: The principle is somewhat similar to that described in "Attention focusing". In the case of "attention focusing" MTL will focus its attention on relevant features to use and for "the feature selection double check" the MTL will help in feature selection. Actually, With a small amount of data and/or significant noise, MTL will be able to better select the relevant feature (inputs) by selecting those that are most significant for all the tasks. That is to say, if one feature is important for more than just one task, then most probably this feature is indeed very important and representative of the data.
5. Representation bias[121]: This is also to help the model generalizes better. Indeed, training MTL for many tasks simultaneously means that we are optimizing 2 loss functions at the same time. That's to say, if the two losses function optimization converge to the same minima, the MTL model will also converge to same minima which might help in reaching better paths.
6. Regularization [115]: With either Statistical data augmentation, Representation bias we are better generalizing the model. Thus we are minimizing the over-fitting risk. We can then consider the MTL as a regulariser since it introducing an inductive bias.

6.1.3 Types of Multi-Task Learning

There are two classical ways to perform Multi-Task learning with deep neural networks: 1) Hard-parameter sharing and 2) Soft-parameter sharing. The main difference between both of them consists in the way the weights are shared inside the neural network architecture [119].

— Hard-parameter Sharing: It is the most commonly used for MTL. It consists in sharing some of the first hidden layers between tasks while keeping several last layers exclusive for different tasks (see figure 6.2).

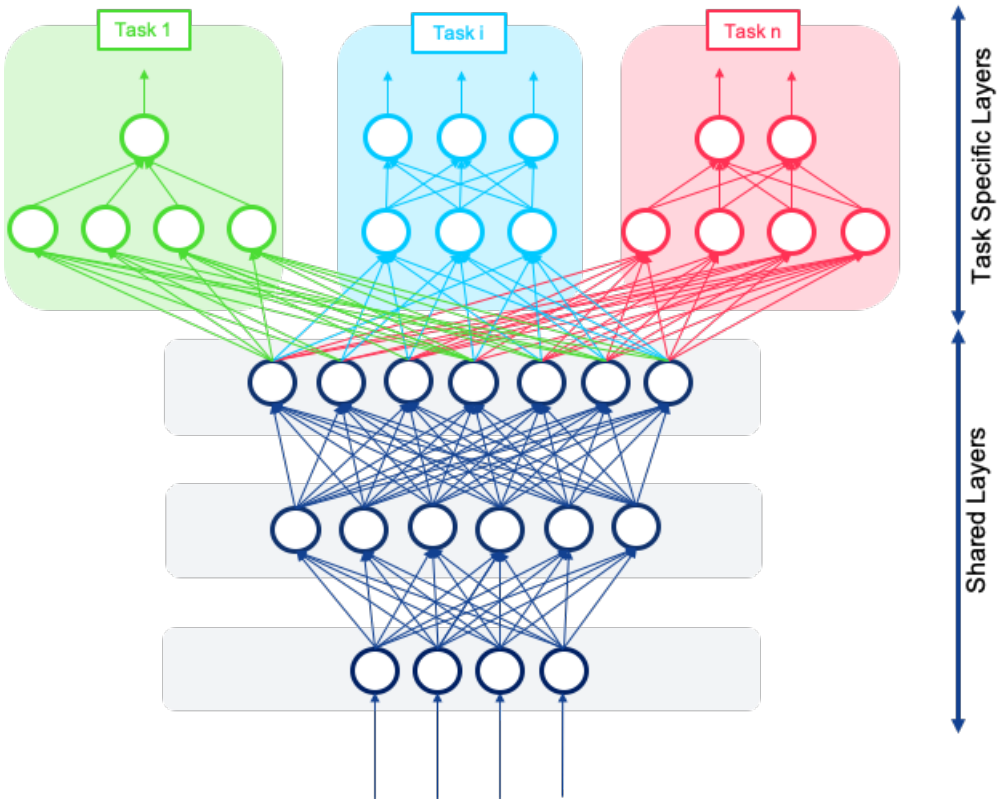

Figure 6.2 – Hard-parameter sharing for Multi-Task learning in deep neuronal Network.

— Soft-parameter Sharing: In soft-parameter Sharing, each task has his own model with its own parameters and setting. This means that there is no physical layer that is shared between the different tasks, but rather

it is an intermediate set of parameters/ or units that is shared. In this case, sharing parameters/ units acts as a weight regulariser [119]. For the sake of explanation, let's assume that we have an MTL problem with two tasks A and B on the same input. A and B are two DL networks that have been trained separately for these tasks. Thus, soft-parameter sharing proposes new units which combine these two networks into a Multi-Task network. This is done in a way such that the tasks supervise how much sharing is needed, as illustrated in Figure 6.3.

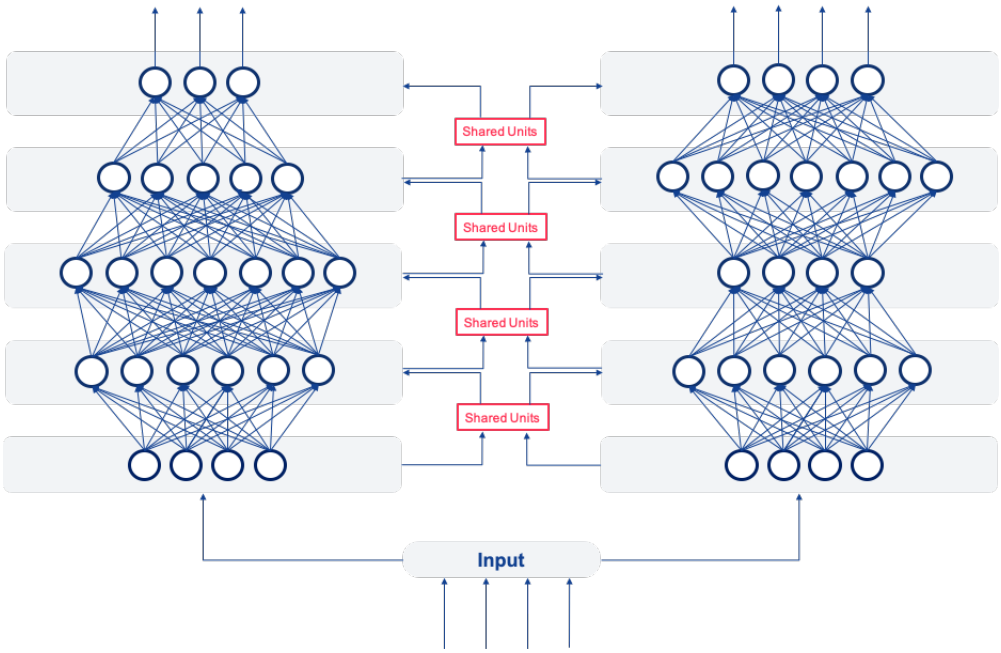

Figure 6.3 – Soft-parameter sharing for Multi-Task learning in deep neuronal Network.

With the success of DL in these last years, many recent works have studied the use of Multi-Task learning with Deep Learning. They proposed many new architectures based on neural networks. Their proposed architectures mainly aim to answer the key questions: What to share and how to learn what has to be shared across tasks? [122] [84]. Although there have been many new proposed architectures to answer what to share and how to share, they all remain based

on classical ones: they all employ the two approaches, we introduced earlier, hard and soft parameter sharing [123].

For soft-parameter sharing, [124] proposed a new architecture called Cross-stitch networks. For their architecture, authors proposed to share features among tasks, by using a linear combination of the activation's found in multiple single task networks. Their model starts with two separate models then a cross-stitch unit is used in order to help the model determine the way, the task-specific networks leverage the knowledge of the other task, by learning a linear combination of the output of the previous layers. Based on works proposed in [124], [125] extended the cross-stitch model and proposed a new architecture called Sluice networks. In [125] and [126], authors propose an architecture that learns what layers and sub-spaces should be shared, as well as at what layers the network has learned the best representations of the input sequences.

The hard-parameter sharing architecture is still the most used and it is the popular one with DL. Hard parameter sharing has been successful in image classification [127], object detection [128],[129], semantic segmentation [130], and facial analysis [128]. Thus, classical architecture that goes back to [115] has been adopted and applied in several fields. It has shown very good performance. Both [131] and [132] present an efficient, real-time, implementation of MTL with hard-parameter sharing architecture that solves three autonomous driving related tasks at once. Their approach builds a branched architecture with a shared encoder, but different decoder branches for each of the three tasks. Like [131] and [132], [133] and [134] have used a MTL architecture with a shared encoder, followed by task-specific decoder networks. Additionally, they both proposed some optimizations to better learn the model. [135] proposes a new architecture called Multi-linear Relationship Network (MRN) for Multi-Task learning, which discovers the task relationships based on multiple task-specific layers of deep convolutional neural networks. That is to say they extend the

classic architecture by placing tensor normal priors on the parameter set of the fully connected layers. In [136], authors propose a dynamic task prioritization for the MTL architecture. They propose to dynamically adjust task-level loss coefficients to continually prioritize difficult tasks. [123] propose a novel approach to decide on the degree of layer sharing between tasks in order to eliminate the need for manual exploration.

6.2 Proposed Multi-Task Learning Architecture

Using MTL to share knowledge among tasks is very useful. However, MTL assumes that the tasks are very similar. If this assumption is not verified then MTL can cause negative transfer and hurt the overall performances. Task relatedness is an open and active research issue [137], [138] [139]. In spite of these early theoretical advances in understanding task relatedness, there isn't much recent progress towards this goal [119], [118]. We still do not have a good notion of when two tasks should be considered similar or related. The classic definition presented in [115] defines two tasks to be similar if they use the same features set or some common features set to make a decision. Based on this definition as well as our prior knowledge, we find that both the tasks of detecting user environment and the user speed range, while consuming a service, are two correlated tasks (as explained in chapter 1), we consider that the assumption of the similarity between our 2 tasks is guaranteed (It will also be proven experimentally).

In general, MTL networks using soft-parameter sharing are limited in terms of scalability, as the size of the network tends to grow linearly with the number of tasks (for N tasks we will have N different Neural Networks). Besides, since we are also constrained by the fact of delivering an answer to the 2 tasks at the same time to limit the synchronization time in the networks (When answers

are delivered in offset) so we opt for a hard-parameter sharing architecture.

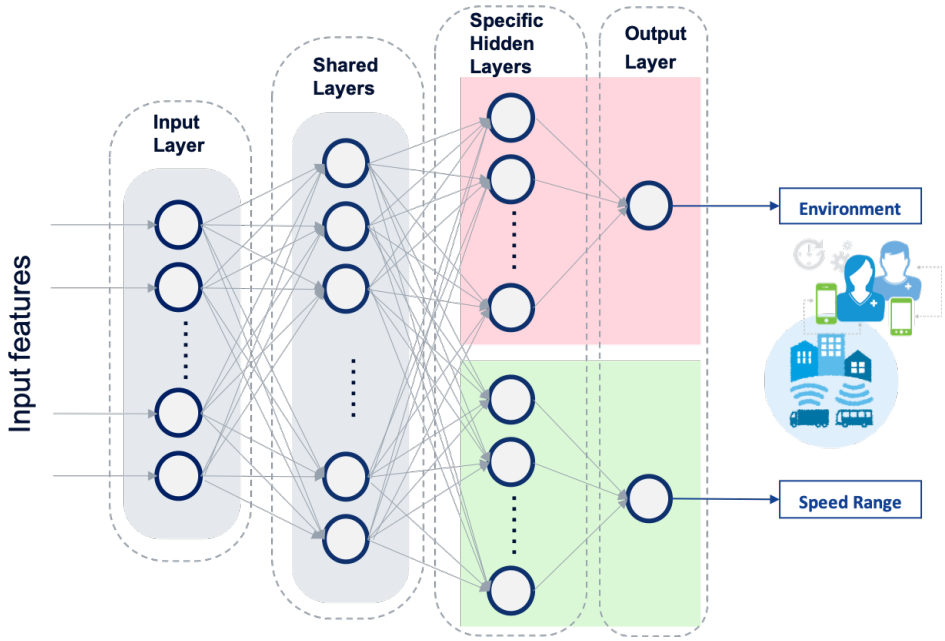

Figure 6.4 – Multi-Task Deep Learning architecture for joint detection of both the user environment and the user speed range simultaneously

Our MTL architecture is based on Deep Learning algorithm (see figure 6.4). The DL choice is explained by the fact that we are dealing with a lot of data. Once deployed this model should deal with thousands or millions of users at the same time. Furthermore, DL is also appropriate for problems where modeling relationships between large number of features are not tractable. This is the case of user behavior detection and more precisely giving a joint and simultaneous answers to both of the user environment and the user speed range while consuming a service. Indeed the model has to extract the complexity and variety of different situations met by mobile users.

6.3 Data features

Based on the definition of [115], which considers two tasks to be similar if they use the same features to make a decision, we note that we use the same features (or inputs) for both the tasks (user environment and user mobility speed range detection). Thus, our feature set is a vector composed of the whole dataset:

— Time: Recording time of signal or burst data arrival (ms).
— RSRP: Average received power of the Reference Signal (RS). The RSRP value lies between -140 dBm to -44 dBm.
— CQI: Channel Quality Indicator that is used to indicate the most appropriate transmission modulation and coding scheme to be used.
— TA: Timing Advance is used to control UL signal transmission timing
— MI: The number of the Cell ID changes (NCID) in a sliding window of a given duration (TCRmax).
— ST: Sojourn time in a cell.
— Extra Signals (feature engineered): signals derived from $RSRP$ and TA.
— Environment Labels: for 8 environments: Home, Work, Mall, Pedestrian, Car, Train, Bus.
— Mobility Labels: for 8 speed categories.
— GPS measurements: they serve to automatically label the data used for Deep Learning training.

This collected data comes from the same large crowd-sourced campaign (described in both chapters 1 and 3).

6.4 Performances: results and discussion

Architecture and configuration

The dataset volume used to empirically test the MTL performance is set to $77k$. This data volume presents $S_{UED--labeled} \cap S^{*}_{MSP--labeled}$ where $S^{*}_{MSP--labeled}$ is extracted from the $S_{MSP--labeled}$ from chapter 5 after data cleaning. In this case the data cleaning consists on dropping instances where the error probability is high. To quantify the error probability of one measurement we use a new metric "GPS-on" to check at first if the measurement was reported with the GPS turned on. Thus the training is done using a part of the labeled data of our dataset. For training, we used (70%) of the labeled data and we used the (30%) remaining for the model performance evaluation.

Hyperparameter optimization was conducted via the Bayesian approach. The set of these hyperparameters is summarized in the table 6.1. We have optimized the model with the best model of UED and MSP in single task. That is to say that for the user environment detection task, we fix our output according to the scheme “4CI_1” (4 Classes [Home, Buildings, Pedestrian, Transport]) (see chapter 4). For the mobility speed profiling task, we fix our output according to the classification scheme $[0, 10[$, $[10, 90[$, $[90, \infty[$ (see chapter 5).

Optim.	layers	Drop.	Valid.	Epoch	Batch	Act.
Adamax	4 shared	0,2	30%	100	100	tanh

Table 6.1 – MTL hyperparameter: The optimizer, the number of hidden layers, the dropout ratio, the number of epoch and the batch size.

As proved experimentally in appendix B, tuning the hyperparameters for one scheme and using the set of the resulting hyperparameters for the other schemes (instead of running an optimization per scheme) allows to save computational time. The performance losses between these 2 scenarios are minimal (around 0.2% of decrease in F1-score) compared to the gain in terms of calcu-

lation time. Thus, in this chapter we will optimize the hyperparameters for one classification scheme. Then, the obtained hyperparameter set will be used to test other schemes.

Results

Tables 6.2 and 6.3 present performance results obtained respectively for the two classification schemes and for the four classification schemes, performed using a single task (Sgt) and a Multi-Task learning (MTL) architecture.

We run experiments on best models for the user environment presented in chapter 4 and user mobility speed range presented in chapter 5.

IOD SgT		MSP Sgt		IOD MTL		MSP MTL	
Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-s.	Acc.	F1-s.
95.80%	95.76%	98.36%	94.25%	**99.12%**	**99.03%**	**97.85%**	**95.25%**

Table 6.2 – Indoor Outdoor detection and mobility speed detection (3 Classes $\{0, 10, 90\}$) classification schemes when performed as a single task (Sgt) or using a Multi-Task learning (MTL) architecture.

UED SgT		MSP Sgt		UED MTL		MSP MTL	
Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-s.	Acc.	F1-s.
94.31%	94.28%	98.36%	94.25%	**99.22%**	**98.23%**	**98.17%**	**96.17%**

Table 6.3 – User Environment detection (4 Classes [Home, Buildings, Pedestrian, Transport]) and mobility speed detection (3 Classes $\{0, 10, 90\}$) classification schemes when performed as a single task (Sgt) or using a Multi-Task learning (MTL) architecture.

As we can see in Table 6.2 and 6.3, UED and MSP detection performance increases with the Multi-Task architecture. We observe also that the performance gain for UED and MSP tasks, respectively, is approximately 3.5% and 1% of F1-score in case of UED with 2 classes. And the gain increases with approximately 4% and 2% of F1-score in case of UED with 4 classes.

The joint learning in a hard-parameter architecture turns out beneficial for both tasks. This is because with such structure they have shared data inputs which brought meaningful additional information to each others. For example, a user can't be indoor with medium or high speed. This performance increase from SgT to MTL also proves experimentally that the 2 tasks that we are trying to solve are similar and correlated. Thus, we have experimentally proven tasks' relatedness (we are referring to UED and MSP tasks).

With MTL, we have reached our success criteria set to 95% of F1-score and we also managed to simultaneously detect two user attributes: the environment as well as the mobility. Comparing these results with our reference case with UED as a binary classification (IOD: Indoor Outdoor Detection) and the same MSP scheme ($\{0, 10, 90\}$), we notice that more granularity in the environment classes information has helped MSP. With MTL based 4 UED classes, MSP detection has improved with 1%. With such promising results, we therefore investigate the performance with more granularity. For MSP SgT, classification schemes that showed best results are (see chapter 5):

— $3C$: $\{0, 10, 90\}$ **kmph** $\longrightarrow [0, 10[, [10, 90[, [90, \infty[$

— $4C$: $\{0, 1, 10, 90\}$ **kmph** $\longrightarrow [0, 1[, [1, 10[, [10, 90[, [90, \infty[$

— $5C$: $\{0, 1, 10, 40, 90\}$ **kmph** $\longrightarrow [0, 1[, [1, 10[, [10, 40[, [40, 90[, [90, \infty[$

— $8C$: $\{0, 1, 2, 3, 10, 30, 40, 90\}$ **kmph** $\longrightarrow [0, 1[, [1, 2[, [2, 3[, [3, 10[, [10, 30[, [30, 40[, [40, 90[, [90, \infty[$

As for the UED SgT, classification schemes that showed best results are (see chapter 4) $2C$, $4CI_0$ and $4CI_1$. Since we are looking for more granularity, we have decided to study up to 8 classes. Thus, UED classification scheme that we are considering in this chapter are as follows:

— $4CI_1$: two 4 classes schemes have been tested in 4: $4CI_0$, $4CI_1$. Since $4CI_1$ was better than $4CI_0$, we keep it as the UED classification scheme for 4 classes.

— $5C_1$: ["Home", "Building", "Mall", "Pedestrian", "Transport"]: in "buildings" namely contains "Work" and various "Buildings" and "Transport" include "Bus", "Train" and "Car".

— $8C$: ["Home", "Work", "Building", "Mall", "Pedestrian", "Bus", "Train", "Car"].

Scheme combination of UED SgT and MSP SgT leads us to the following classification schemes for our MTL model:

1. $MSP_{3C} - UED_{4CI_1}$:
   — MSP: $\{0, 10, 90\}$ kmph
   — UED: ["Home", "Building", "Pedestrian", "Transport"]
2. $MSP_{3C} - UED_{5C_1}$:
   — MSP: $\{0, 10, 90\}$ kmph
   — UED: ["Home", "Building", "Mall", "Pedestrian", "Transport"]
3. $MSP_{3C} - UED_{8C}$:
   — MSP: $\{0, 10, 90\}$ kmph
   — UED: ["Home", "Work", "Building", "Mall", "Pedestrian", "Bus", "Train", "Car"]
4. $MSP_{4C} - UED_{4CI_1}$:
   — MSP: $\{0, 1, 10, 90\}$ kmph
   — UED: ["Home", "Building", "Pedestrian", "Transport"]
5. $MSP_{4C} - UED_{5C_1}$:
   — MSP: $\{0, 1, 10, 90\}$ kmph
   — UED: ["Home", "Building", "Mall", "Pedestrian", "Transport"]
6. $MSP_{4C} - UED_{8C}$:
   — MSP: $\{0, 1, 10, 90\}$ kmph
   — UED: ["Home", "Work", "Building", "Mall", "Pedestrian", "Bus", "Train", "Car"]
7. $MSP_{5C} - UED_{4CI_1}$:

— MSP: $\{0, 1, 10, 40, 90\}$ kmph

— UED: [“Home”, “Building”, “Pedestrian”, “Transport”]

8. $MSP_{5C} - UED_{5C_1}$:
   — MSP: $\{0, 1, 10, 40, 90\}$ kmph
   — UED: [“Home”, “Building”, “Mall”, “Pedestrian”, “Transport”]
9. $MSP_{5C} - UED_{8C}$:
   — MSP: $\{0, 1, 10, 40, 90\}$ kmph
   — UED: [“Home”, “Work”, “Building”, “Mall”, “Pedestrian”, “Bus”, “Train”, “Car”]
10. $MSP_{8C} - UED_{4CI_1}$:
    — MSP: $\{0, 1, 2, 3, 10, 30, 40, 90\}$ kmph
    — UED: [“Home”, “Building”, “Pedestrian”, “Transport”]
11. $MSP_{8C} - UED_{5C_1}$:
    — MSP: $\{0, 1, 2, 3, 10, 30, 40, 90\}$ kmph
    — UED: [“Home”, “Building”, “Mall”, “Pedestrian”, “Transport”]
12. $MSP_{8C} - UED_{8C}$:
    — MSP: $\{0, 1, 2, 3, 10, 30, 40, 90\}$ kmph
    — UED: [“Home”, “Work”, “Building”, “Mall”, “Pedestrian”, “Bus”, “Train”, “Car”]

The detailed results of these MTL schemes are presented in table 6.4. These results have been obtained with the same set of hyperparameters which we got when we ran Bayesian search to tune the hyperparameters in table 6.3. Table 6.4 presents the accuracy and F1-score metrics for both UED-MTL and MSP-MTL. To better analyse the results, we also introduce a metric which is the mean of both the accuracy and F1-score for UED-MTL and MSP-MTL. MTL is beneficial for both tasks: as we notice in the figure 6.7 F1-score for both UED and MSP is higher with MTL architecture (dotted line) than when detected as a single task (continuous line).

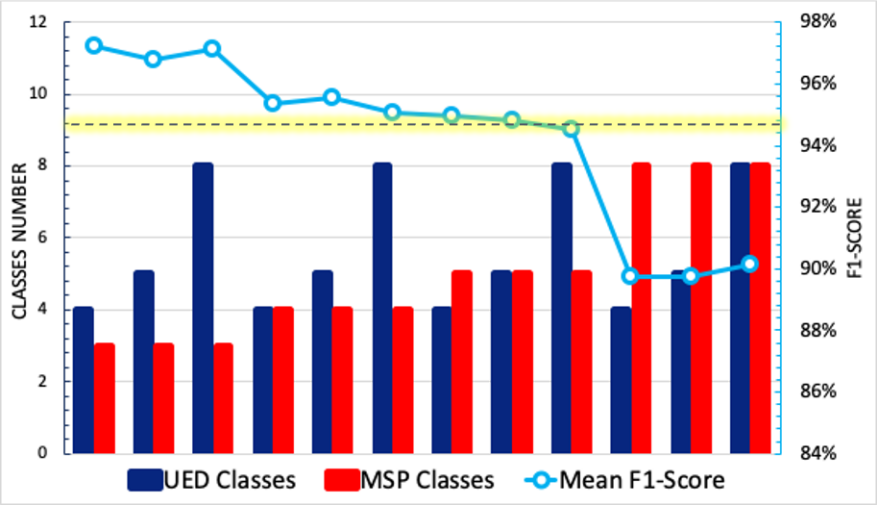

Figure 6.5 – The variation of classes number per task vs the mean F1-score delivered by the MTL model

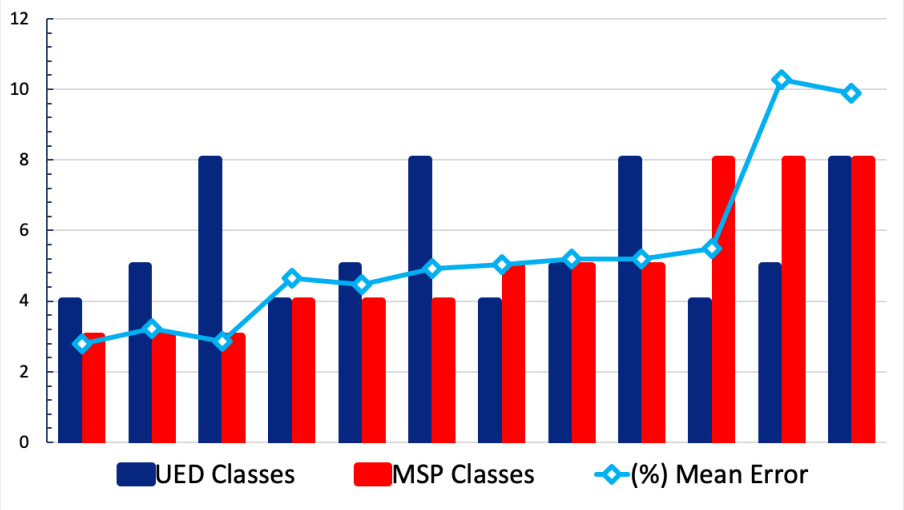

Figure 6.6 – The variation of classes number per task vs the mean error (1-F1-score) delivered by the MTL model

The error analysis (1 - F1-score) obtained with respect to the number of classes for UED and MSP tasks (see table 6.4 and figures 6.5 and 6.6)), shows at first that the overall performance of MTL is promising. Most of the studied schemes show a mean F1-score around or higher than 95%. Increasing the number of UED classes does not affect that much the overall performance of

the system. For example when we set the MSP classification scheme at 3C, we notice an error variance in UED of 0.3% of F1score. The worst UED-MTL performance, for the classification scheme $MSP_{8C} - UED_{8C}$, shows 97.37% as F1-score which satisfies our success criteria. On the other hand, the increase in MSP classes' number negatively affects the overall performance. The more the number of classes is increased, the more the performance decreases (see figure 6.7). In fact, increasing granularity in MSP increases the probability of the label noise in the data which is very harmful to the classification task.

The 2 best scenarios delivering the best performance are: 1) MSP_{3C}–UED_{4CI_1} (MSP: $\{0, 10, 90\}$ kmph and UED: "Home", "Building", "Pedestrian", "Transport"]) and 2) MSP_{3C}–UED_{8C} (MSP: $\{0, 10, 90\}$ kmph and UED: ["Home", "Work", "Building", "Mall", "Pedestrian", "Bus", "Train", "Car"]). For these two we note a mean F1-score 97.20% and 97.14%, respectively.

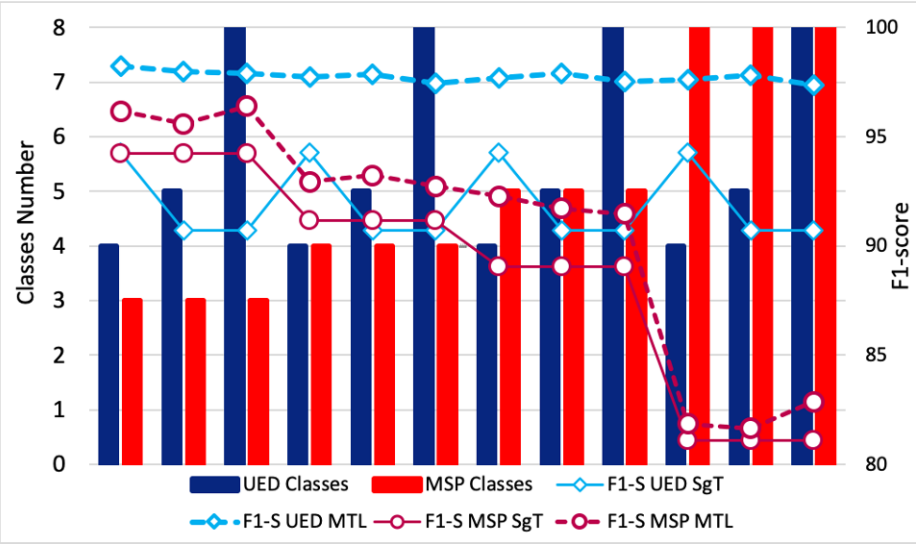

Figure 6.7 – The variation of classes number per task vs the F1-score delivered by the MTL model for both tasks: MSP & UED

	$MSP_{3C} - UED_{4CI_1}$		$MSP_{3C} - UED_{5C_1}$		$MSP_{3C} - UED_{8C}$		$MSP_{4C} - UED_{4CI_1}$	
	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.
UED MTL	99.22%	98.23%	98.92%	97.97%	98.85%	97.90%	98.90%	97.75%
MSP MTL	98.17%	96.17%	97.93%	95.60%	98.23%	96.39%	95.19%	92.93%
Mean MTL perf.	98.69%	97.20%	98.42%	96.78%	98.54%	97.14%	97.04%	95.34%
	$MSP_{4C} - UED_{5C_1}$		$MSP_{4C} - UED_{8C}$		$MSP_{5C} - UED_{4CI_1}$		$MSP_{5C} - UED_{5C_1}$	
	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.
UED MTL	98.96%	97.84%	98.74%	97.44%	98.88%	97.68%	98.90%	97.90%
MSP MTL	95.35%	93.22%	95.10%	92.72%	94.93%	92.27%	94.51%	91.72%
Mean MTL perf.	97.15%	95.53%	96.92%	95.08%	96.90%	94.97%	96.70%	94.81%
	$MSP_{5C} - UED_{8C}$		$MSP_{8C} - UED_{4CI_1}$		$MSP_{8C} - UED_{5C_1}$		$MSP_{8C} - UED_{8C}$	
	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.	Acc.	F1-S.
UED MTL	98.74%	97.54%	98.85%	97.61%	98.86%	97.82%	98.66%	97.37%
MSP MTL	94.44%	91.46%	92.21%	81.85%	92.37%	81.64%	92.60%	82.87%
Mean MTL perf.	96.59%	94.50%	95.53%	89.73%	95.61%	89.73%	95.63%	90.12%

Table 6.4 – User Environment detection and mobility speed detection classification schemes when performed using a Multi-Task learning (MTL) architecture and the mean MTL performance over the two tasks.

6.5 Conclusion

In this chapter, we have presented the basics of the Multi-Task Learning (MTL) which is a subcategory of transfer Learning (TL). One of the advantages of MTL is the benefit obtained by sharing the knowledge between two or more

tasks while learning them simultaneously. Thus, we have investigated MTL to jointly learn both the user environment detection (UED) and mobility speed profiling (MSP), while consuming a service. We have proposed an intelligent system for MSP detection linked with user activity preference. We used hard-parameter sharing between the layers of the neural networks where the first layers are commonly shared between the two tasks followed by some task specific layers. Based on crowd-sourced user-specific data, we achieved more than 97% of overall F1-score performance. This result satisfies our success criteria of at least obtaining 95% of F1-score which is very encouraging to investigate the user behavior.

Chapter 7

Conclusion & Perspectives

With all the innovations and the new services brought by 5G such as Internet of Things (IoT), autonomous cars, Device to device (D2D) communications, etc., many new challenges are raised for a better management and orchestration of network and services. To face these challenges, a fully end-to-end automation is desired for 5G networks and beyond to largely manage themselves and deal with organisation, configuration, security, and optimisation issues. Such automation will allow different components of the mobile network to operate autonomously without external intervention. From the user side, who is always seeking for a better service, it is expected from the future generation networks (5G networks and beyond) to better accommodate the ever-growing user-demands, services and applications and will guarantee a better Quality of Experience (QoE) (QoE is a metric that measures the mobile user's satisfaction depending on his or her experience of a mobile service). One way to guarantee this is by adding in-advance network context-awareness capabilities. Thus the networks can be aware of the user preferences or habits while consuming a service (or using their phones). The use of this external knowledge based on the user habits will make the networks efficiently face the variable consuming habits of users that impact the network conditions. It will make them more intelligent and more aware of the environment in which they operate. However, inferring the user behavior based on his preferences, remains a complex problem since we are dealing with multi composite environments, many services

and billions of users.

In this dissertation, our first contribution was to propose a 3-step system that automatically extracts the knowledge of the mobile user behavior in order to anticipate the user preferences, and needs, with the final goal to optimize 5G networks. The 3-step system was based on machine learning and artificial intelligence approaches. The first step deals with data collection, the second tackles the data processing for user behavior inference and user profile building and the third and final step is the use of information for 5G network optimization. Artificial Intelligence (AI) can be seen as an efficient solution to bring the required intelligence to 5G networks by helping them to be aware of the consuming habits or preferences of users and then to help them anticipating users' actions or needs. For this, in this thesis, we investigated Machine Learning and Deep Learning techniques and data analysis in order to infer the mobile user behavior.

The second contribution of this thesis was to model the user behavior. We defined a user behavior model based on the abstraction of contextual information of a mobile user, which was defined by answering 6 main questions (Who, Where, What, When, How, Why). The contextual information was made of four parts covering the use context, the application, demographics and the device. Each answer to a question above corresponded to an attribute related to mobile user's contextual information. These attributes were also Quality-of-Experience(QoE)-influencing features. Then, we presented an investigation of user behavior modeling with two of the QoE-influencing features related to mobile use context of contextual information: the environment and the user mobility speed range. The user's environment and mobility have a big influence on QoE. For example, a user who is indoor would experience a very different service quality as compared to the one who is outdoor, all else being equal. Actually, the environment and the mobility together set up the conditions in

which a mobile user consumes the requested services/applications. In practice, in order to estimate the environment or mobility, we had to answer the following questions: how and where a mobile user consumes the mobile services? Besides, these questions had to be answered at the same time (simultaneously) to detect and then predict the user behavior.

Providing a mathematical model to answer these questions simultaneously (how and where) is a way too complex due to huge number of users and the variety of different situations met by mobile users. Our third contribution was thus to infer the mobile user behavior reflecting the real life of user (the environment as well as the mobility range of the user), with minimal human intervention, low-complexity system and minimal response time. Thus, our proposed solution was based on Machine learning approaches and more precisely Deep Learning approaches. The Multi-Task based Deep Learning architecture was adopted to provide a joint answer to these two questions simultaneously. At first, we investigated the two tasks separately. Then, we investigated the Multi-Task architecture for joint detection.

For every machine learning or Deep Learning approach, data can be seen as the fuel or the corner stone for the success of these approaches. Without relevant and a representative data, the machine learning approaches are limited. We proposed in our work to collect data based on a crowd-sourcing mode. In this mode, we involved the crowd in the collection process which provided data reflecting better the users' behaviors. Machine Learning algorithms trained on datasets collected in crowd-sourcing mode allow to learn very diverse real-world situations. However, this collection mode generates noisy data, since it is not strictly controlled during the data collection. This noise added to the imbalanced data, inherited from the user behavior who prefers to use his phone in some situations than in others, remains a challenge to deal with. Thus, as another contribution in this thesis, we proposed some methods and algorithms

to deal with such noisy and imbalanced datasets.

7.1 The mobile user environment detection while consuming a service

7.1.1 Conclusions

The first work of this thesis was to study the first attribute for the user behavior detection which deals with the user environment. To simplify, the user environment had been considered as a binary classification problem, referred as IOD: Indoor Outdoor Detection. It was inferred using a ML approach based on 3GPP input data known at the network side: $\{Time, RSRP, CQI, MI, TA\}$. Compared to the state of the art, adding both of the mobility indicator MI and the timing advance TA, and replacing the quality signal $RSRQ$ by an other quality signal CQI, had shown better results than using only the couple $(RSRP, RSRQ)$. For a further investigation, we focused on detecting the user environment with more granularity than just two classes: Indoor/ Outdoor. That is to say with more detailed classes of the user environment (going up to 5 classes). The choice of the number of classes was based on the preliminary analysis of the signals $\{Time, RSRP, CQI, MI, TA\}$ vs. the user behavior. We studied how to divide the initial indoor/outdoor classes further into more detailed environment types respecting a trade-off between the data instances per class and the granularity degree by varying the number of environment classes. Thus, we studied different combinations of splitting the environment into different classes and compared their performance. The best performance for the multi-class detection was delivered by the 4-classes classification scheme splitting the class indoor in two classes "Home" and other "Buildings" and splitting the class outdoor in two classes "Pedestrian" and "Transport".

Secondly, in order to also exploit the unlabeled data, a comparative study of semi-supervised approaches was conducted. We began by a comparative study of three semi-supervised approaches for the case of 2-classes detection (Indoor or Outdoor). These three semi-supervised approaches were: 1) cluster then label 2) co-training and 3) self-training. Results showed that the best approach was the self-training since it is less greedy in resources and also delivering better accuracy than the two others. Thus, we also studied the self-training for the multi-class user environment detection. Results showed that, using the unlabeled data with the self-training enhanced the overall performances by 2% on average, whether for the first case (for indoor outdoor detection) or the second (for the multi-class detection).

The study of semi-supervised approaches raised our curiosity to investigate the minimum ratio of labeled data compared to unlabeled data, while guaranteeing a good performance. Thus, we investigated the impact of the volume of labeled data or unlabeled data used on the overall performances for the indoor/ outdoor detection. While maintaining same target performance, we could reduce up to 30% the amount of labeled data. Such an evaluation - namely the required ratio between labeled and unlabeled data for a target IOD performance - could be of interest to operators. Avoiding to tag all data, reduces the labeling efforts and constraints for the operators wanting to implement indoor outdoor algorithm. Self-Training can thus perform well without requiring a complete labeling of data.

7.1.2 Perspectives

To go further, the work can be extended by exploring the user activity and time correlation since time is directly linked to user behavior. The time correlation can be investigated via Long short-term memory (LSTM) Deep Learning architecture, or via recurrent neural network (RNN) Deep Learning. These two

architectures can enhance the overall performance of the user environment detection. However a comparative study for a tradeoff between performance and time should be conducted since LSTM and RNN are more complex and resource greedy than FNN. Another future idea is to study the transfer learning approaches in order to quickly adapt the learnt model from our data in case of new coming data.

7.2 The user mobility speed profile detection while consuming a service

7.2.1 Conclusions

The second work in this thesis was to investigate the second attribute of the user behavior: mobility speed profile detection while consuming a service. So far, in the state of the art, the user mobility speed profile detection had been considered as three speed profiles (Low, Medium, High). For these 3 states the speed borders were set as $[0, 40[$kmph, $[40, 90[$kmph and $[90, \infty[$kmph, as it is advised in literature. An analysis on the dataset collected showed that these borders were not the best reflecting the behavior of user studied, while consuming a service when jointly considering the environment and speed. Indeed the borders closely depend on the real life activity of users. Thus we expanded our study by investigating the user mobility and speed profile detection using a supervised Deep Learning algorithm, based on a multi-output classification.

The main issues that we faced with the mobility investigations were: the noise in the labels and the imbalanced classes. Actually, in order to compute the label (the ground truth) for the mobility detection, we used GPS coordinates which were very noisy. To overcome this issue we proposed solutions to clean these labels. As for the imbalanced class issue, we employed the Artificial Data

Augmentation technique.

7.2.2 Perspectives

In future, we would like to more investigate the noise in the labels of the mobility detection task. During this work, some solutions were proposed to solve this issue. Despite the gain brought by these solutions, they remain limited. The noise in the labels is much more important to be covered by these solutions. That's why we propose to investigate new solutions to overcome this issue. For example, to use totally unsupervised methods or to use the transfer learning to transfer the knowledge between the first attribute to the second attribute. An other idea also to optimize the mobility detection performances is to explore the time correlation since time is directly linked to user mobility.

7.3 Joint detection of the user environment and mobility speed profile while consuming a service

7.3.1 Conclusions

Our last contribution in this thesis, was to investigate the joint detection of both attributes of user behavior: the user environment and the user mobility speed range while consuming a service. The simultaneous detection of both attributes was achieved with good accuracy thanks to a Multi-Task Deep Learning architecture. One of the advantages of such architecture is the benefit obtained by sharing the knowledge between two or more tasks while learning them simultaneously. The architecture we proposed is based on sharing the first layers of neural networks between the two attributes and then to setting

up of some specific layers for each task.

At first, we investigated the Multi-Task detection for two simple cases: 1) case one: the environment detection task as 2 classes indoor or outdoor, and the mobility detection task as a 3 classes detection $\{0, 10, 90\}$ 2) case two: the environment detection task as 4 classes: Home, Buildings, Pedestrian, In-Transport and the mobility detection task as a 3 classes detection $\{0, 10, 90\}$. Results for these two cases were very promising. For a further investigation, we focused on user behavior Multi-Task detection with more granularity going up to 8 classes for both of the user environment detection as well as the user mobility speed profile detection.

7.3.2 Perspectives

In future, we would like to explore the integration of the other attributes modeling the user behavior in the Multi-Task architecture. Furthermore, we would like to quantify the impact of the user behavior detection on the quality of experience (QoE).

Appendices

Appendix A

Hyperparameter Tuning Methods Comparison

To experimentally prove what we have discussed theoretically in chapter 3, we propose in this section to experimentally study the performance of 3 hyperparameter optimization methods: i) manual search, ii) Grid search and iii) Bayesian search.

A.1 Mobile user Environment Detection

To quantify the impact of the optimization method, we fix the classification scheme to be $4CI\_1$. Now, if we fix the number of the hidden layers to 7, the training step of the neural network takes in average 130.86 second. The challenge is to find the best combination of the hyperparameters that can learn the best UED classifier. For optimization, we consider 7 hyperparameters: the epoch size, the batch size, the activation function, the initialization, the learning rate, the dropout rate, the optimizer, and the number of neurons per layer. Different values of these hyperparameters and their combinations are very large. This means that the convergence time to train the best model is also very high. Let us assume that we loop among 5 possibilities or values of the epoch size, 5 possibilities of the batch size, 3 possibilities of the activation function, 3 possibilities of the initialization, 3 possibilities of the learning rate, 5 possi-

bilities of the dropout rate, 5 possibilities of the optimizer, and 4 possibilities of the number of neurons per layer. This leads us to test 472500 neural networks and it would take around $472500 * 130.86\ seconds$ for them to train. To limit the run time, we investigated 3 methods: manual search, grid search and Bayesian search as shown in figure A.1. We set a threshold of 50 iterations for both the Bayesian optimization and the grid search (an iteration = one combination of hyperparameter for training a model). As expected the manual search is the fastest one since it is conducted by a human (10 runs at most). As shown in figure A.1 and supported by analysis before, the Grid search is too slow and the computation time is high. In the worst case, the best model is delivered after looping on all combinations of hyperparameters. Whereas, Bayesian method finds the right range and parameters space from the first iterations.

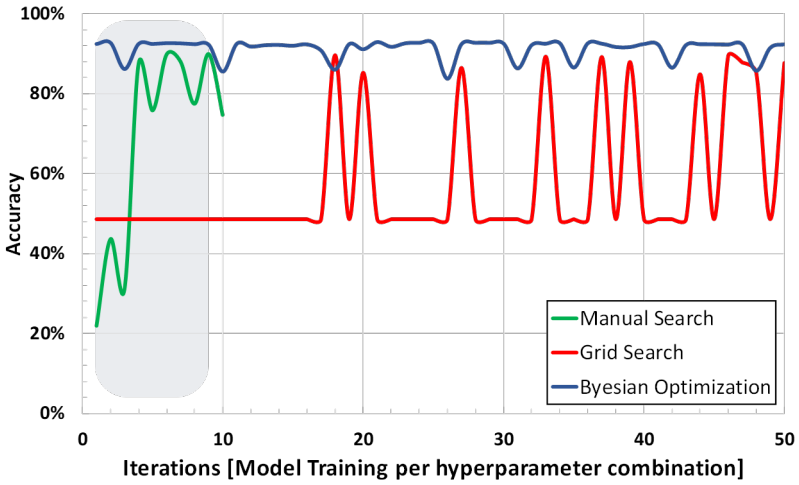

Figure A.1 – Impact of optimization methods on the accuracy computing -User Environment Detection - Classification scheme $4CI\_1$

A.2 User Mobility Speed Profiling

Like what has been described in the the previous section, we fix the classification scheme $\{0, 10, 90\}$ kmph (the one that has delivered the best perfor-

mances). Figure A.2 confirms that again the Bayesian optimisation for neuronal networks outperforms both of the two other tested approaches: grid search and manual search.

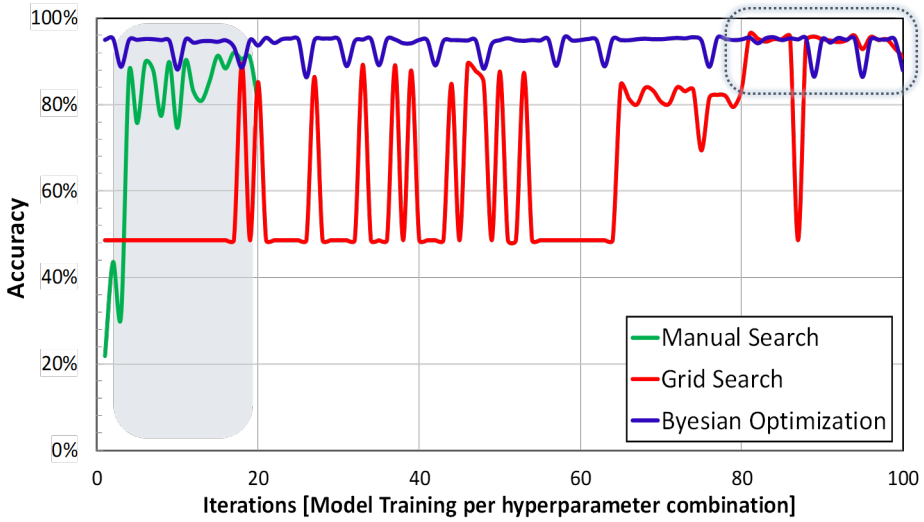

Figure A.2 – Impact of optimization methods on the accuracy computing - User Mobility seed profile detection - classification scheme $\{0, 10, 90\}$ kmph

Appendix B

Hyperparameter Tuning: Tuning one model for many classification schemes study

---

The research space for the hyperparameter's setting is infinite. An expert knowledge is needed in order to fix a finite space that optimization methods will use to find the best hyperparameters for a given task. The biggest this space is the longer the computing time will be for the different optimizations methods.

In this appendix we try to quantify the impact of optimizing one classification scheme and to use the set of the best hyperparameters to study other classification schemes. For this we fix the classification scheme to be $4CI\_1$ of the UED (User environment detection task. We run a Bayesian optimization over this scheme and the resulting hyperparameters are used to evaluate other classification schemes.

Hyperparameter	Possibilities
Activation	$[tanh, ReLu]$
Optimizer	$[nadam, rmsprop, adam, adagrad]$
Dropout Rate	$[0.2, 0.3, 0.4]$

Table B.1 – Research space of hyperparameters for the $4CI\_1$ UED (User Environment Detection) classification scheme

The finite research space that we have fixed for the scheme $4CI\_1$ is de-

scribed in table B.1. For such a research space, the running time to get optimal hyperparameters is $16680.3531611s$ ($\approx 04:38:01$ in $HH:MM:SS$) using a 2.3 GHz 8-Core Intel Core i9 as a CPU and 32 GB of memory. This running time could have been much larger if we increased the search space (knowing that we fixed the number of epoch and the number of batch to respectively 70 and 200). The best hyperparameters set resulting from the Bayesian optimisation search is described in the table B.2.

Optim.	layers	Dropout	Valid.	Epoch	Batch	Act.
Adam	7	0,2	20%	70	200	tanh

Table B.2 – Best hyperparameter's set for the $4CI\_1$ UED (User Environment Detection) classification scheme. The Hyperparameter's list is: The optimizer, the number of hidden layers, the dropout ratio, the number of epoch and the batch size.

Now, we try to quantify the performances of some different UED classification schemes while using the same hyperparameters set found in the table B.3 for the $4CI\_1$ UED (User Environment Detection) classification scheme. We choose 2 schemes with less class numbers ($3C_0$ and $3C_1$) and 2 schemes with a higher class number ($5C_0$ and $5C_1$) and one scheme with the same class number ($4CO_0$). Such choice is made in order to quantify the model behavior in case of less complex problem (3 classes), a more complex problem (5 classes) and a problem presenting the same complexity (4 classes). As we can notice using the hyperparameter set of the scheme $4CI\_1$ does not hurt drastically the F1-score. We note an average of 0.112 F1- score decrease.

The gain obtained by optimizing each model for each classification scheme compared to the computing time for the hyperparamleters search is insignificant. Thus in case of many classification schemes, it's better to run an optimization search over one model and then use the resulting hyperparameters set for other schemes.

Classification scheme	Performances with Optim		Performances without Optim	
	Acc.	F1-Sc.	Acc.	F1-Sc.
$3C_0$	91.89%	91.84%	91.85%	91.81%
$3C_1$	92.06%	92.04%	92.00%	$92,02\%$
$4CO_0$	91.58%	91.53%	91.42%	91.42%
$5C_0$	90.70%	90.69%	90.67%	90.60%
$5C_1$	90.79%	90.72%	90.49%	90.41%

Table B.3 – Performances of different UED classification schemes with Bayesian optimisation for each scheme vs using the hyperparameters set resulting from a Bayesian optimisation run on one classification scheme (the $4CI\_1$ UED (User Environment Detection) classification scheme).

Appendix C

Résumé en Français

Apprentissage Automatique Pour Déduire le Comportement des Utilisateurs Dans les Réseaux Autonomes 5G

Au cours des quatre dernières décennies, les réseaux mobiles ont connu quatre générations différentes. Avec chaque génération de nouvelle technologies, des améliorations, des nouveautés ou de nouveaux services sont apparus [2]. Aujourd'hui, la nouvelle cinquième génération (5G) des réseaux mobiles suscite beaucoup d'intérêt. En effet, cette génération a promis de révolutionner les réseaux de communication mobile. Parmi les nouveautés que propose la 5G on peut citer l'Internet des Objets (IoT), les véhicules autonomes, la communication Device to Device (D2D) et pleins d'autres. Toutes ces nouveautés obligent à une meilleure gestion et orchestration des réseaux 5G et de ses services. Pour relever ces défis, une automatisation de bout en bout est préconisée

pour les réseaux 5G afin de gérer les problèmes d'organisation, de configuration, de sécurité et d'optimisation. Une telle optimisation permettera alors aux différents composants du réseau mobile d'opérer d'un manière autonome sans l'intervention extérieure de quoi ou de qui que ce soit.

D'un point de vue utilisateur mobile, qui est toujours à la recherche de la meilleure qualité de service et à moindre coût, cette nouvelle génération doit donc satisfaire ses besoins et lui garantir une meilleure qualité d'experience (QoE). La QoE est une métrique qui mesure le niveau de satisfaction de l'utilisateur mobile en fonction de son expérience d'un service mobile. Une manière de guarantir la satisfaction de l'utilisateur est d'injecter une connaissance suplémentaire basée sur le comportement de l'utilisateur dans les réseaux 5G. En effet, les réseaux mobiles enrichis avec cette connaissance suplémentaire peuvent être conscients des préférences ou des habitudes des utilisateurs quand ils consomment un service (ou quand ils utilisent leurs téléphones mobiles). Cette connaissance des préférences des utilisateurs permettera aux réseaux de faire face aux habitudes de consommation variables des utilisateurs qui ont un impact sur les conditions du réseau. Ce qui engendrera des réseaux plus intelligents qui peuvent se gérer en grande partie et gérer de façon automatique les problèmes d'organisation, de configuration, de sécurité et d'optimisation. Cependant, inférer le comportement d'utilisateur est consideré comme un problème complexe compte tenu des multiples environements dans lesquels le réseau opère et le grand nombre d'utilisateurs (de l'ordre de billions).

Dans cette thèse, on propose un système à trois étapes afin de permettre l'extraction des informations liées au comportement de l'utilisateur et de les injecter dans les réseaux mobiles. Ce système est basé sur des technologies et des méthodes d'intelligence artificielle. Ces 3 étapes sont : 1) la collecte des données 2) le traitement des données pour l'inférence du comportement des utilisateurs et la construction des profils des utilisateurs et finalement 3) l'utilisation de

cette information pour l'optimisation. Cette thèse traite principalement des 2 premières étapes. On s'est basés sur des approches d'intelligence artificielle et plus spécifiquement sur l'apprentissage automatique et l'apprentissage profond (Deep Learning). En effet, ces approches peuvent être considérées comme des solutions efficaces pour apporter l'intelligence requise aux réseaux 5G et pour les aider à connaitre les habitudes de consommation ou les préférences des utilisateurs, pour prédire leurs besoins et anticiper les actions à mettre en oeuvre pour offrir la QoE souhaitée tout en minimisant les ressources. Elles sont aussi très efficaces pour la modélisation mathématique d'un problème complexe, ce qui est le cas de l'inférence du comportement des utilisateurs.

Plusieurs modélisations du comportement des utilisateurs ont été etudiées dans la littérature, mais la plupart de ces études ont consideré une seule composante ou un seul attribut du comportement d'utilisateur. Or en réalité, le comportement d'utilisateur est plus complexe et ne peut être considéré que via un seul attribut. Dans cette thèse, on modélise le comportement comme une abstraction d'un large nombre de situations dans lesquelles un utilisateur mobile consomme ou utilise des services (ou des applications mobiles variés) et ayant un impact sur la QoE. Afin de définir le modèle du comportement de l'utilisateur en tenant compte de ces exigences, on définit ce modèle en répondant à 6 questions basées sur la méthode de Kipling [29]. Ces questions sont (qui, où, quoi, quand, comment, pourquoi) et leurs réponses détaillées et spécifiques vont permettre une analyse complète du problème. En outre, chaque réponse à une question correspond à un attribut lié aux informations contextuelles des mobiles qui sont des fonctionnalités influençant la qualité d'expérience (QoE).

On présente ensuite une étude investigant la modélisation du comportement des utilisateurs avec deux des fonctionnalités influençant le plus la QoE : l'environnement de l'utilisateur mobile quand il consomme un service et la

nature (ou le type) de mobilité matérialisée par sa vitesse de déplacement. L'environnement et la mobilité de l'utilisateur ont une grande influence sur la QoE. Par exemple, un utilisateur qui est à l'intérieur (indoor) connaîtrait une qualité de service différente de celle qu'il peut avoir s'il est à l'extérieur (outdoor). En effet, l'environnement et la mobilité définissent ensemble les conditions dans lesquelles un utilisateur mobile consomme les services / applications souhaités. Par rapport à notre modélisation du comportement d'utilisateur, l'estimation de l'environnement ou de la mobilité, répond aux questions suivantes : comment et où un utilisateur mobile consomme les services mobiles? En outre, ces questions doivent être répondues simultanément pour détecter puis prédire le comportement de l'utilisateur.

Pour inférer le comportement de l'utilisateur mobile (l'environnement ainsi que la l'intervalle de sa mobilité), avec le minimum d'intervention humaine et avec un système de faible complexité, on propose dans cette thèse une solution basée sur un aprenstissage profond multi-tâches. L'architecture d'apprentissage profond basée sur le multi-tâches a été adoptée pour fournir une réponse conjointe à ces deux questions et simultanément. Dans un premier temps, on a étudié les deux tâches séparément. Ensuite, on a étudié l'architecture multi-tâches pour la détection conjointe.

Les résultats pour les deux solutions proposées (la detection de l'environement et la detection de l'intervalle de vitesse) ainsi que la détection conjointe des deux tâches simultanément sont très prometteurs. Ce qui nous amène à croire à l'intérêt de l'utilisation de la cognition sur le comportement de l'utilisateur dans les réseaux mobiles. Cette cognition aidera les réseaux 5G à faire face aux habitudes de consommation variables des utilisateurs qui ont un impact sur les conditions du réseau. En même temps, les demandes ou les besoins des clients seront satisfaits individuellement.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] F. Long, N. Li, and Y. Wang, « Autonomic mobile networks: The use of artificial intelligence in wireless communications », *in 2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, Aug. 2017, pp. 582–586. DOI: 10.1109/ICARM.2017.8273227.

[3] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, « Networks and devices for the 5G era », *IEEE Communications Magazine*, vol. 52, *2*, pp. 90–96, 2014.

[4] S. S. Mwanje and C. Mannweiler, « Towards Cognitive Autonomous Networks in 5G », *in 2018 ITU Kaleidoscope: Machine Learning for a 5G Future (ITU K)*, IEEE, 2018, pp. 1–8.

[5] S. van der Meer, J. Keeney, and L. Fallon, « 5G networks must be autonomic! », *in NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2018, pp. 1–5.

[6] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, « A survey of autonomic network architectures and evaluation criteria », *IEEE Communications Surveys & Tutorials*, vol. 14, *2*, pp. 464–490, 2011.

[7] G. Pujolle, *Les réseaux*. Editions Eyrolles, 2014.

[8] N. Chairmen of ISG ENI MEC and ZSM, « ETSI White Paper No. 32: Network Transformation; (Orchestration, Network and Service Management Framework) », Tech. Rep., Oct. 2019. [Online]. Available: https://

www.etsi.org/images/files/ETSIWhitePapers/ETSI_White_Paper_Network_Transformation_2019_N32.pdf.

[9] C. Benzaid and T. Taleb, « AI-driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions », *IEEE Network*, vol. 34, *2*, pp. 186–194, 2020.

[10] « Zero-touch network and Service Management (ZSM); Reference Architecture », Tech. Rep., Aug. 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf.

[11] E. Bulut and B. K. Szymanski, « Understanding user behavior via mobile data analysis », *in 2015 IEEE International Conference on Communication Workshop (ICCW)*, IEEE, 2015, pp. 1563–1568.

[12] S. Maaloul, M. Afif, and S. Tabbane, *A new vertical handover decision based context awareness for ubiquitous access*, IEEE, 2012.

[13] B. Han, S. Wong, C. Mannweiler, M. R. Crippa, and H. D. Schotten, « Context-awareness enhances 5G multi-access edge computing reliability », *IEEE Access*, vol. 7, pp. 21 290–21 299, 2019.

[14] B. O. Holzbauer, B. K. Szymanski, and E. Bulut, « Impact of socially based demand on the efficiency of caching strategy », *in 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, IEEE, 2014, pp. 401–406.

[15] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, « Identifying diverse usage behaviors of smartphone apps », *in Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ACM, 2011, pp. 329–344.

[16] D. Wu, Q. Wu, Y. Xu, J. Jing, and Z. Qin, « QoE-based distributed multi-channel allocation in 5G heterogeneous cellular networks: A matching-coalitional game solution », *IEEE Access*, vol. 5, pp. 61–71, 2016.

[17] T. Soikkeli, « The effect of context on smartphone usage sessions », *Aalto University School of Science*, 2011.

[18] D. Peraković, I. Forenbacher, I. Jovović, *et al.*, « IDENTIFICATION AND PREDICTION OF USER BEHAVIOR DEPENDING ON THE CONTEXT OF THE USE OF SMART MOBILE DEVICES. », *Annals of DAAAM & Proceedings*, vol. 26, *1*, 2015.

[19] T. Soikkeli, J. Karikoski, and H. Hammainen, « Diversity and end user context in smartphone usage sessions », *in 2011 Fifth International Conference on Next Generation Mobile Applications, Services and Technologies*, IEEE, 2011, pp. 7–12.

[20] S. Mekki, T. Karagkioules, and S. Valentin, « HTTP adaptive streaming with indoors-outdoors detection in mobile networks », *in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2017, pp. 671–676.

[21] A. Ray, S. Deb, and P. Monogioudis, « Localization of LTE measurement records with missing information », *in IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, IEEE, 2016, pp. 1–9.

[22] A. Ulvan, M. Ulvan, and R. Bestak, « The enhancement of handover strategy by mobility prediction in broadband wireless access », *in Proceedings of the networking and electronic commerce research conference (NAEC 2009)*, American Telecommunications Systems Management Association Inc., 2009, pp. 266–276.

[23] C. Wang, Z. Zhao, Q. Sun, and H. Zhang, « Deep Learning-based Intelligent Dual Connectivity for Mobility Management in Dense Network », *in 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, IEEE, 2018, pp. 1–5.

[24] M.-L. A. Morel and S. Randriamasy, « Quality of experience-aware enhanced inter-cell interference coordination for self organized HetNet », *in 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC)*, IEEE, 2017, pp. 1–8.

[25] S. Cicalo, N. Changuel, V. Tralli, B. Sayadi, F. Faucheux, and S. Kerboeuf, « Improving QoE and fairness in HTTP adaptive streaming over LTE network », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, *12*, pp. 2284–2298, 2015.

[26] S. ISO, « 13407: 1999 », *Human-centred design processes for interactive systems*, 1999.

[27] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, « Towards a better understanding of context and context-awareness », *in International symposium on handheld and ubiquitous computing*, Springer, 1999, pp. 304–307.

[28] V. A. Siris, K. Balampekos, and M. K. Marina, « Mobile quality of experience: Recent advances and challenges », *in 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, IEEE, 2014, pp. 425–430.

[29] N. Javaid, A. Sher, H. Nasir, and N. Guizani, « Intelligence in IoT-based 5G networks: Opportunities and challenges », *IEEE Communications Magazine*, vol. 56, *10*, pp. 94–100, 2018.

[30] S. Liu, J. Mcgree, Z. Ge, and Y. Xie, *Computational and statistical methods for analysing big data with applications*. Academic Press, 2015.

[31] C. Zhang, P. Patras, and H. Haddadi, « Deep learning in mobile and wireless networking: A survey », *IEEE Communications Surveys & Tutorials*, vol. 21, *3*, pp. 2224–2287, 2019.

[32] Y. LeCun, Y. Bengio, and G. Hinton, « Deep learning », *Nature*, vol. 521, *7553*, pp. 436–444, 2015.

[33] M. K. Marina, V. Radu, and K. Balampekos, « Impact of indoor-outdoor context on crowdsourcing based mobile coverage analysis », *in Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, 2015, pp. 45–50.

[34] S. Ickin, K. Wac, M. Fiedler, L. Janowski, J.-H. Hong, and A. K. Dey, « Factors influencing quality of experience of commonly used mobile applications », *IEEE Communications Magazine*, vol. 50, *4*, pp. 48–56, 2012.

[35] D. Lodder, B. Bregen, D. Braunschweig, J. Allen-Smith, and T. Ford. (2015). « Understanding the Right Mobile Mix for Your Venue », [Online]. Available: `https://www.hetnetforum.com/resources/send/2-resources/36-venueconnect-2015-hetnet-forum-presentation`.

[36] T. Soikkeli, J. Karikoski, and H. Hammainen, « Diversity and end user context in smartphone usage sessions », *in 2011 Fifth International Conference on Next Generation Mobile Applications, Services and Technologies*, IEEE, 2011, pp. 7–12.

[37] E. Kaasinen, *User acceptance of mobile services: Value, ease of use, trust and ease of adoption*, 2005.

[38] P. Branco, L. Torgo, and R. Ribeiro, « A survey of predictive modelling under imbalanced distributions », *arXiv preprint arXiv:1505.01658*, 2015.

[39] I. Saffar, M. L. A. Morel, K. D. Singh, and C. Viho, « Machine Learning with partially labeled Data for Indoor Outdoor Detection », *in 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2019, pp. 1–8.

[40] I. Saffar, M. L. A. Morel, K. D. Singh, and C. Viho, « Semi-supervised Deep Learning-based Methods for Indoor Outdoor Detection », *in ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–7.

[41] I. Saffar, M. L. A. Morel, M. Amara, K. D. Singh, and C. Viho, « Mobile User Environment Detection using Deep Learning based Multi-Output Classification », *in 2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*, IEEE, 2019, pp. 16–23.

[42] I. Saffar, M. L. A. Morel, K. D. Singh, and C. Viho, « Deep Learning Based Speed Profiling for Mobile Users in 5G Cellular Networks », *in 2019 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2019, pp. 1–7.

[43] M.-L. Alberi-Morel, I. Saffar, K. Singh, and C. Viho, « Multi-task Deep Learning based Environment and Mobility Detection for User Behavior Modeling », 2019.

[44] I. Arel, D. C. Rose, and T. P. Karnowski, « Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier] », *IEEE Computational Intelligence Magazine*, vol. 5, *4*, pp. 13–18, Nov. 2010, ISSN: 1556-603X. DOI: 10.1109/MCI.2010.938364.

[45] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, « Machine Learning Paradigms for Next-Generation Wireless Networks », *IEEE Wireless Communications*, 2016.

[46] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, « Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks », *arXiv preprint arXiv:1710.02913*, vol. 9, 2017.

[47] E. Brynjolfsson and T. Mitchell, « What can machine learning do? Workforce implications », *Science*, vol. 358, *6370*, pp. 1530–1534, 2017.

[48] A. M. Learning, « Developer Guide », *Amazon Web Services*, 2018.

[49] J. Alzubi, A. Nayyar, and A. Kumar, « Machine learning from theory to algorithms: an overview », *in Journal of physics: conference series*, vol. 1142, 2018, p. 012 012.

[50] J. Brownlee, *Master Machine Learning Algorithms: discover how they work and implement them from scratch*. Machine Learning Mastery, 2016.

[51] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[52] P. Preux, « Fouille de données, notes de cours », *Disponible sur internet*, 2007.

[53] K. He, X. Zhang, S. Ren, and J. Sun, « Deep residual learning for image recognition », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[54] W. S. McCulloch and W. Pitts, « A logical calculus of the ideas immanent in nervous activity », *The bulletin of mathematical biophysics*, vol. 5, *4*, pp. 115–133, 1943.

[55] F. Rosenblatt, « The perceptron: A probabilistic model for information storage and organization in the brain. », *Psychological review*, vol. 65, *6*, p. 386, 1958.

[56] B. Widrow and M. E. Hoff, « Adaptive switching circuits », STANFORD UNIV CA STANFORD ELECTRONICS LABS, Tech. Rep., 1960.

[57] L. Deng, « A tutorial survey of architectures, algorithms, and applications for deep learning », *APSIPA Transactions on Signal and Information Processing*, vol. 3, e2, 2014.

[58] P. Ramachandran, B. Zoph, and Q. V. Le, « Swish: a self-gated activation function », *arXiv preprint arXiv: 1710.05941*, vol. 7, 2017.

[59] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for machine learning*. Cambridge University Press, 2020.

[60] E. Charniak, *Introduction to deep learning*. The MIT Press, 2019.

[61] Z. Tupikovskaja-Omovie and D. Tyler, « Mobile consumer shopping journey in fashion retail: eye tracking mobile apps and websites », *in Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, 2018, pp. 1–3.

[62] L. Molyneux, « Mobile news consumption: A habit of snacking », *Digital Journalism*, vol. 6, *5*, pp. 634–650, 2018.

[63] L. Leung and C. Chen, « Extending the theory of planned behavior: A study of lifestyles, contextual factors, mobile viewing habits, TV content interest, and intention to adopt mobile TV », *Telematics and Informatics*, vol. 34, *8*, pp. 1638–1649, 2017.

[64] F. Chollet, *Deep Learning with Python*. Manning, Nov. 2017, ISBN: 978161 7294433.

[65] G. C. M. solution overview. (). « Workflow de machine learning », [Online]. Available: `https://cloud.google.com/ml-engine/docs/ml-solutions-overview`.

[66] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[67] A. Fernández, S. Garcıa, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*. Springer, 2018.

[68] M. Claesen and B. De Moor, « Hyperparameter search in machine learning », *arXiv preprint arXiv: 1502.02127*, 2015.

[69] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, « Algorithms for hyper-parameter optimization », *in Advances in neural information processing systems*, 2011, pp. 2546–2554.

[70] J. Bergstra and Y. Bengio, « Random search for hyper-parameter optimization », *Journal of machine learning research*, vol. 13, *Feb*, pp. 281–305, 2012.

[71] J. Bergstra, D. Yamins, and D. D. Cox, « Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms », *in Proceedings of the 12th Python in science conference*, Citeseer, 2013, pp. 13–20.

[72] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, « Dropout: a simple way to prevent neural networks from overfitting », *The journal of machine learning research*, vol. 15, *1*, pp. 1929–1958, 2014.

[73] J. Sietsma and R. J. Dow, « Creating artificial neural networks that generalize », *Neural networks*, vol. 4, *1*, pp. 67–79, 1991.

[74] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, « Big data-driven optimization for mobile networks toward 5G », *IEEE network*, vol. 30, *1*, pp. 44–51, 2016.

[75] H. Li, X. Xu, C. Liu, T. Ren, K. Wu, X. Cao, W. Zhang, Y. Yu, and D. Song, « A machine learning approach to prevent malicious calls over telephony networks », *in 2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018, pp. 53–69.

[76] D. Tsilimantos, T. Karagkioules, and S. Valentin, « Classifying flows and buffer state for youtube's HTTP adaptive streaming service in mobile networks », *in Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 138–149.

[77] D. Damopoulos, S. A. Menesidou, G. Kambourakis, M. Papadaki, N. Clarke, and S. Gritzalis, « Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers », *Security and Communication Networks*, vol. 5, *1*, pp. 3–14, 2012.

[78] W. A. Hapsari, A. Umesh, M. Iwamura, M. Tomala, B. Gyula, and B. Sebire, « Minimization of drive tests solution in 3GPP », *IEEE Communications Magazine*, vol. 50, *6*, pp. 28–36, 2012.

[79] U. T. R. Access, « and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio measurement collection for Minimaztion of Drive Test (MDT), 3GPP », Technical Specification, Technical report TS 37.320, Tech. Rep., 2014.

[80] A. Faggiani, E. Gregori, L. Lenzini, V. Luconi, and A. Vecchio, « Smartphone-based crowdsourcing for network monitoring: opportunities, challenges, and a case study », *IEEE Communications Magazine*, vol. 52, *1*, pp. 106–113, 2014.

[81] J. Cainey, B. Gill, S. Johnston, J. Robinson, and S. Westwood, « Modelling download throughput of LTE networks », *in 39th Annual IEEE Conference on Local Computer Networks Workshops*, IEEE, 2014, pp. 623–628.

[82] I. Misra and L. van der Maaten, « Self-supervised learning of pretext-invariant representations », *arXiv preprint arXiv: 1912.01991*, 2019.

[83] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, « Multiple instance learning: A survey of problem characteristics and applications », *Pattern Recognition*, vol. 77, pp. 329–353, 2018.

[84] Y. Zhang and Q. Yang, « A survey on multi-task learning », *arXiv preprint arXiv: 1707.08114*, 2017.

[85] K. Konyushkova, R. Sznitman, and P. Fua, « Learning Active Learning from Data », *in Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 4225–4235. [Online]. Available: http://papers.nips.cc/paper/7010-learning-active-learning-from-data.pdf.

[86] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, « Online learning: A comprehensive survey », *arXiv preprint arXiv: 1802.02871*, 2018.

[87] N. Kato, Z. M. Fadlullah, B. Mao, F. Tang, O. Akashi, T. Inoue, and K. Mizutani, « The Deep Learning Vision for Heterogeneous Network Traffic Control: Proposal, Challenges, and Future Perspective », *IEEE Wireless Communications*, vol. PP, *99*, pp. 2–9, 2017, ISSN: 1536-1284. DOI: 10.1109/MWC.2016.1600317WC.

[88] V. Radu, P. Katsikouli, R. Sarkar, and M. K. Marina, « A semi-supervised learning approach for robust indoor-outdoor detection with smartphones », *in Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014, pp. 280–294.

[89] A. Ng, « Nuts and bolts of building AI applications using Deep Learning », *NIPS Keynote Talk*, 2016.

[90] S. Edelev, S. N. Prasad, H. Karnal, and D. Hogrefe, « Knowledge-assisted location-adaptive technique for indoor-outdoor detection in e-learning »,

in 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), IEEE, 2015, pp. 8–13.

[91] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen, « Iodetector: A generic service for indoor outdoor detection », *in Proceedings of the 10th acm conference on embedded network sensor systems*, 2012, pp. 113–126.

[92] A. B. H. Alaya-Feki, A. Le Cornec, and E. Moulines, « Optimization of Radio Measurements Exploitation in Wireless Mobile Networks. », *JCM*, vol. 2, 7, pp. 59–67, 2007.

[93] M. Haddad, D. G. Herculea, E. Altman, N. B. Rached, V. Capdevielle, C. S. Chen, and F. Ratovelomanana, « Mobility state estimation in LTE », *in 2016 IEEE Wireless Communications and Networking Conference*, IEEE, 2016, pp. 1–6.

[94] D. Herculea, C. S. Chen, M. Haddad, and V. Capdevielle, « Straight: Stochastic geometry and user history based mobility estimation », *in Proceedings of the 8th ACM International Workshop on Hot Topics in Planet-scale mObile computing and online Social neTworking*, 2016, pp. 1–6.

[95] *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode*, Standard, 2018.

[96] *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification*, Standard, 2018.

[97] *Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*, Standard, 2018.

[98] *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*, Standard, 2018.

[99] *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*, Standard, 2018.

[100] A. O. Laiyemo, « High speed moving networks in future wireless systems. », Ph.D. dissertation, University of Oulu, Finland, 2018.

[101] X. Zhu and A. B. Goldberg, « Introduction to semi-supervised learning (synthesis lectures on artificial intelligence and machine learning) », *Morgan and Claypool Publishers*, vol. 14, 2009.

[102] X. J. Zhu, « Semi-supervised learning literature survey », University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[103] Z.-H. Zhou and M. Li, « Tri-training: Exploiting unlabeled data using three classifiers », *IEEE Transactions on knowledge and Data Engineering*, vol. 17, *11*, pp. 1529–1541, 2005.

[104] X. Zhu, J. Lafferty, and R. Rosenfeld, « Semi-supervised learning with graphs », Ph.D. dissertation, Carnegie Mellon University, language technologies institute, school of . . ., 2005.

[105] H.-W. Lee, N.-r. Kim, and J.-H. Lee, « Deep neural network self-training based on unsupervised learning and dropout », *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, *1*, pp. 1–9, 2017.

[106] S. Michaelis, *Balancing high-load scenarios with next cell predictions and mobility pattern recognition*. Citeseer, 2012.

[107] A. Ulvan, M. Ulvan, and R. Bestak, « The enhancement of handover strategy by mobility prediction in broadband wireless access », *in Proceedings of the networking and electronic commerce research conference (NAEC 2009)*, American Telecommunications Systems Management Association Inc., 2009, pp. 266–276.

[108] C. Wang, Z. Zhao, Q. Sun, and H. Zhang, « Deep Learning-based Intelligent Dual Connectivity for Mobility Management in Dense Network », *in 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, IEEE, 2018, pp. 1–5.

[109] X. Ge, J. Ye, Y. Yang, and Q. Li, « User mobility evaluation for 5G small cell networks based on individual mobility model », *IEEE Journal on Selected Areas in Communications*, vol. 34, *3*, pp. 528–541, 2016.

[110] S. Banville and F. V. Diggelen. (Nov. 2016). « Innovation: Precise positioning using raw GPS measurements from Android smartphones », [Online]. Available: https://www.gpsworld.com/innovation-precise-positioning-using-raw-gps-measurements-from-android-smartphones/.

[111] G. Developers. (2019). « Optimize location for battery », [Online]. Available: https://developer.android.com/guide/topics/location/battery.

[112] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, « SMOTE: synthetic minority over-sampling technique », *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[113] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, « Understanding data augmentation for classification: when to warp? », *in 2016 international conference on digital image computing: techniques and applications (DICTA)*, IEEE, 2016, pp. 1–6.

[114] *Study on scenarios and requirements for next generation access technologies*, Standard, 2019.

[115] R. Caruana, « Multitask Learning: A Knowledge-Based Source of Inductive Bias ICML », *Google Scholar Google Scholar Digital Library Digital Library*, 1993.

[116] S. J. Pan and Q. Yang, « A survey on transfer learning », *IEEE Transactions on knowledge and data engineering*, vol. 22, *10*, pp. 1345–1359, 2009.

[117] J. Baxter, R. Caruana, T. Mitchell, L. Y. Pratt, D. L. Silver, and S. Thrun, *NIPS 1995 workshop on learning to learn: Knowledge consolidation and transfer in inductive systems*.

[118] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, « Transfer learning using computational intelligence: A survey », *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[119] S. Ruder, « An overview of multi-task learning in deep neural networks », *arXiv preprint arXiv: 1706.05098*, 2017.

[120] Y. S. Abu-Mostafa, « Learning from hints in neural networks », *Journal of complexity*, vol. 6, *2*, pp. 192–198, 1990.

[121] J. Baxter, « A Bayesian/information theoretic model of learning to learn via multiple task sampling », *Machine learning*, vol. 28, *1*, pp. 7–39, 1997.

[122] E. Meyerson and R. Miikkulainen, « Beyond shared hierarchies: Deep multitask learning through soft layer ordering », *arXiv preprint arXiv: 1711.00108*, 2017.

[123] S. Vandenhende, S. Georgoulis, B. De Brabandere, and L. Van Gool, « Branched multi-task networks: deciding what layers to share », *arXiv preprint arXiv: 1 904.02920*, 2019.

[124] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, « Cross-stitch networks for multi-task learning », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3994–4003.

[125] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, « Latent multi-task architecture learning », *in Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4822–4829.

[126] S. Liu, E. Johns, and A. J. Davison, « End-to-end multi-task learning with attention », *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1871–1880.

[127] H. Bilen and A. Vedaldi, « Integrated perception with recurrent multi-task neural networks », *in Advances in neural information processing systems*, 2016, pp. 235–243.

[128] R. Ranjan, V. M. Patel, and R. Chellappa, « Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, *1*, pp. 121–135, 2017.

[129] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, « Focal loss for dense object detection », *in Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[130] K. He, G. Gkioxari, P. Dollár, and R. Girshick, « Mask r-cnn », *in Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[131] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, « Fast scene understanding for autonomous driving », *arXiv preprint arXiv: 1708. 02550*, 2017.

[132] A. Kendall, Y. Gal, and R. Cipolla, « Multi-task learning using uncertainty to weigh losses for scene geometry and semantics », *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[133] O. Sener and V. Koltun, « Multi-task learning as multi-objective optimization », *in Advances in Neural Information Processing Systems*, 2018, pp. 527–538.

[134] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, « Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks », *arXiv preprint arXiv: 1711.02257*, 2017.

[135] M. Long, Z. Cao, J. Wang, and S. Y. Philip, « Learning multiple tasks with multilinear relationship networks », *in Advances in neural information processing systems*, 2017, pp. 1594–1603.

[136] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, « Dynamic task prioritization for multitask learning », *in Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 270–287.

[137] C. Shui, M. Abbasi, L.-É. Robitaille, B. Wang, and C. Gagné, « A principled approach for learning task similarity in multitask learning », *arXiv preprint arXiv: 1903.09109*, 2019.

[138] G. Strezoski, N. van Noord, and M. Worring, « Learning task relatedness in multi-task learning for images in context », *in Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 2019, pp. 78–86.

[139] S. Wu, H. R. Zhang, and C. Ré, « Understanding and Improving Information Transfer in Multi-Task Learning », *arXiv preprint arXiv: 2005.00944*, 2020.

Titre : Apprentissage automatique pour déduire le comportement des utilisateurs dans Réseaux autonomes 5G

Mot clés : LTE, 5G, Deep learning, Indoor/Outdoor detection (IOD), Mobility Speed Profile (MSP) Detection

Résumé : L'idée développée dans cette thèse est d'utiliser le Machine Learning/Deep Learning et l'analyse de données radio 3GPP pour estimer et prédire le comportement d'un utilisateur, en termes d'habitudes et de préférence d'usage des services mobiles d'un réseau 5G. Le caractère multidimensionnel du comportement de l'utilisateur rend son estimation complexe et reste actuellement un défi. On a donc étudié son estimation sous une approche innovante au regard de l'état de l'art. On a proposé de la réaliser au sein d'un système unifié qui estime en parallèle chaque dimension du comportement. En utilisant des méthodes basées sur l'apprentissage approfondi (deep-learning) supervisé et hybride/semi-supervisé, on propose une solution pour la détection de l'environnement (Indoor/Outdoor Detection (IOD)) et jusqu'à 8 classes d'environnement d'un utilisateur de téléphone portable. Nous proposons ensuite une solution permettant de détecter la catégorie de mobilité (Mobility Speed Profile (MSP) Detection) jusqu'à 8 profils de vitesses. Enfin, une solution innovante basée sur des algorithmes d'apprentissage profond dans une architecture multitâches permet d'estimer conjointement à la fois l'environnement et le profil de mobilité. La comparaison avec l'état de l'art a montré l'efficacité des méthodes proposées. Ce qui permet d'envisager leur utilisation par des opérateurs mobiles au sein de leurs futurs.

Title: Machine Learning to Infer User Behavior in 5G autonomic networks

Keywords: LTE, 5G, Deep learning, Indoor/Outdoor detection (IOD), Mobility Speed Profile (MSP) Detection

Abstract: The main idea of this thesis is to use machine learning/deep learning techniques to estimate and predict user behavior by analyzing 3GPP radio signals. The user behavior is defined in terms of habits and preferences while consuming 5G services. The estimation of user behavior is complex and remains a challenge due to its multidimensional nature. We therefore studied an innovative approach for the user behavior estimation: we use a unified system which jointly as well as parallelly estimates each dimension of behavior. Using methods based on supervised and hybrid / semi-supervised deep-learning, we propose a solution for the detection of the user environment (from Indoor / Outdoor Detection (IOD) to up to 8 classes). We then propose a solution to detect the mobility categories (Mobility Speed Profile (MSP) Detection) up to 8 speed profiles. Finally, an innovative solution jointly estimates both the environment and the mobility profile using deep learning algorithms and a multitasking architecture. The comparison with the state of the art shows the effectiveness of the proposed methods. This allows to consider its deployment by operators in future.