Artificial Intelligent based Multi Agent Control Systems

Federico Lisi

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE,
INFORMATICA E STATISTICA
PhD Program in Automatica, Bioengineering and Operation Research
CURRICULUM IN AUTOMATICA

Advisor
Chiar.mo Prof. Francesco Delli Priscoli
Candidate
Ing. Federico Lisi

DIAG - DIPARTIMENTO DI INGEGNERIA INFORMATICA, AUTOMATICA E GESTIONALE ANTONIO RUBERTI
A.A. 2017-2018

I would like to dedicate this thesis to my great love... LARA

Declaration

I, Federico Lisi, hereby declare that this PhD thesis, titled "Artificial Intelligent based Multi Agent Control Systems (Sistemi di Controllo Multi Agent basati su tecniche di Intelligenza Artificiale)", and the work in it are my own. I confirm that:

1. this work has been done mainly during the research activities performed in my own PhD carrier at the University of Rome "La Sapienza".
2. some part of this PhD thesis was presented in MIUR project and H2020 project
3. some part of this PhD thesis are already presented, submitted and accepted in international publication for both Journal and Conference.
4. where I have consulted other published works they are always cited.
5. where I have quoted from the work of others, the source is always given and, with the exception of such quotations, this thesis is entirely my own work
6. the co-authors of my published works are thanked in proper section of this PhD thesis

In faith,

Federico Lisi

Inscription

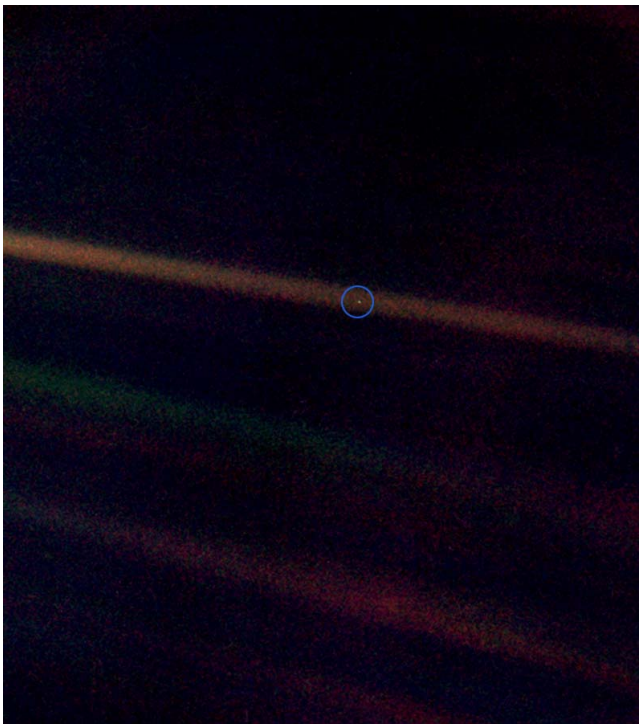

Fig. 1 Pale Blue Dot, Carl Segan

"We succeeded in taking that picture [from deep space], and, if you look at it, you see a dot. That's here. That's home. That's us. On it, everyone you ever heard of, every human being who ever lived, lived out their lives. The aggregate of all our joys and sufferings, thousands of confident religions, ideologies and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilizations, every king and peasant, every young couple in love, every hopeful child, every mother and father, every inventor and explorer, every teacher of morals, every corrupt politician, every superstar, every supreme leader, every saint and sinner in the history of our species, lived there on a mote of dust, suspended in a sunbeam.
The Earth is a very small stage in a vast cosmic arena. Think of the rivers of

blood spilled by all those generals and emperors so that in glory and in triumph they could become the momentary masters of a fraction of a dot. Think of the endless cruelties visited by the inhabitants of one corner of the dot on scarcely distinguishable inhabitants of some other corner of the dot. How frequent their misunderstandings, how eager they are to kill one another, how fervent their hatreds. Our posturings, our imagined self-importance, the delusion that we have some privileged position in the universe, are challenged by this point of pale light.
Our planet is a lonely speck in the great enveloping cosmic dark. In our obscurity – in all this vastness – there is no hint that help will come from elsewhere to save us from ourselves. It is up to us. It's been said that astronomy is a humbling, and I might add, a character-building experience. To my mind, there is perhaps no better demonstration of the folly of human conceits than this distant image of our tiny world. To me, it underscores our responsibility to deal more kindly and compassionately with one another and to preserve and cherish that pale blue dot, the only home we've ever known."

"Da questo distante punto di osservazione, la Terra può non sembrare di particolare interesse. Ma per noi, è diverso. Guardate ancora quel puntino. È qui. È casa. È noi. Su di esso, tutti coloro che amate, tutti coloro che conoscete, tutti coloro di cui avete mai sentito parlare, ogni essere umano che sia mai esistito, hanno vissuto la propria vita. L'insieme delle nostre gioie e dolori, migliaia di religioni, ideologie e dottrine economiche, così sicure di sé, ogni cacciatore e raccoglitore, ogni eroe e codardo, ogni creatore e distruttore di civiltà, ogni re e plebeo, ogni giovane coppia innamorata, ogni madre e padre, figlio speranzoso, inventore ed esploratore, ogni predicatore di moralità, ogni politico corrotto, ogni "superstar", ogni "comandante supremo", ogni santo e peccatore nella storia della nostra specie è vissuto lì, su un minuscolo granello di polvere sospeso in un raggio di sole. La Terra è un piccolissimo palco in una vasta arena cosmica.
Pensate ai fiumi di sangue versati da tutti quei generali e imperatori affinché, nella gloria e nel trionfo, potessero diventare per un momento padroni di una frazione di un puntino. Pensate alle crudeltà senza fine inflitte dagli abitanti di un angolo di questo pixel agli abitanti scarsamente distinguibili di qualche altro angolo, quanto frequenti le incomprensioni, quanto smaniosi di uccidersi a vicenda, quanto fervente il loro odio. Le nostre ostentazioni, la nostra immagi-

naria autostima, l'illusione che noi abbiamo una qualche posizione privilegiata nell'Universo, sono messe in discussione da questo punto di luce pallida. Il nostro pianeta è un granellino solitario nel grande, avvolgente buio cosmico. Nella nostra oscurità, in tutta questa vastità, non c'è alcuna indicazione che possa giungere aiuto da qualche altra parte per salvarci da noi stessi.
La Terra è l'unico mondo conosciuto che possa ospitare la vita. Non c'è altro posto, per lo meno nel futuro prossimo, dove la nostra specie possa migrare. Visitare, sì. Colonizzare, non ancora. Che ci piaccia o meno, per il momento la Terra è dove ci giochiamo le nostre carte. È stato detto che l'astronomia è un'esperienza di umiltà e che forma il carattere. Non c'è forse migliore dimostrazione della follia delle vanità umane che questa distante immagine del nostro minuscolo mondo. Per me, sottolinea la nostra responsabilità di occuparci più gentilmente l'uno dell'altro, e di preservare e proteggere il pallido punto blu, l'unica casa che abbiamo mai conosciuto."

Carl Segan, October 13, 1994

Acknowledgements

I would like to express my deepest gratitude to my PhD Supervisor, Professor Francesco Delli Priscoli, for believing in me and for being a working guide. I will always carry all his teachings with me.
I would like to thank Dr. Silvia Canale for having trained me professionally, for having teaching me how to work internationally, for helping me in the difficulty encountered during my work, and for being close to me until at the end of this trip.
I would like to thank Professor Antonio Pietrabissa, the nicest and most serious person I've ever met. I thank him for helping me in developing some part of this work, and for being a special working and life guide. A special thank to my friends and colleagues Raffaele Gambuti, Alessandro Giuseppi and Martina Panfili for the time spent together.
I would like to thank all my colleagues Vincezo Suraci, Alessandro Di Giorgio, Letterio Zuccaro, Lorenzo Ricciardi Celsi, Federico Cimorelli, Francesco Liberati, Antonio Ornatelli and Andrea Tortorelli

Lastely, I would like to thank my parents for helping me during these years, I would like also thank my brother, and my grandparents.
This work was being possible also thanks to the love of my life, LARA.

Desidero esprimere la mia più profonda gratitudine al Professore Francesco Delli Priscoli, per aver creduto in me e per essere stato la guida durante lo svolgimento di questo lavoro. Porterò sempre con me tutti i suoi insegnamenti.
Vorrei ringraziare la Dottoressa Silvia Canale per avermi formato professionalmente, per avermi insegnato come lavorare a livello internazionale, per avermi aiutato nella difficoltà durante il lavoro e per essermi stata vicino fino alla fine di questo viaggio.

Vorrei ringraziare il Professore Antonio Pietrabissa, la persona più bella e seria che abbia mai incontrato. Lo ringrazio per avermi aiutato nello sviluppo di una parte di questo lavoro e per essere stato una speciale guida di vita e di lavoro. Un ringraziamento speciale al miei amici e colleghi Raffaele Gambuti, Alessandro Giuseppi e Martina Panfili.
Vorrei ringraziare tutti i miei colleghi Vincezo Suraci, Alessandro Di Giorgio, Letterio Zuccaro, Lorenzo Ricciardi Celsi, Federico Cimorelli, Francesco Liberati, Antonio Ornatelli e Andrea Tortorelli.

Infine, mi piacerebbe ringraziare i miei genitori per avermi aiutato in questi anni, grazie anche a mio fratello e ai miei nonni.
Questo lavoro è stato realizzato anche grazie all'amore della mia vita LARA.

Abstract

Artificial Intelligence (AI) is a science that deals with the problem of having machines perform intelligent, complex, actions with the aim of helping the human being. It is then possible to assert that Artificial Intelligence permits to bring into machines, typical characteristics and abilities that were once limited to human intervention. In the field of AI there are several tasks that ideally could be delegated to machines, such as environment aware perception, visual perception and complex decisions in various field.
The recent research trends in this field have produced remarkable upgrades mainly on complex engineering systems such as multi-agent systems, networked systems, manufacturing, vehicular and transportation systems, health care; in fact, a portion of the mentioned engineering system is discussed in this PhD thesis, as most of them are typical field of application for traditional control systems.
The main purpose if this work is to present my recent research activities in the field of complex systems, bringing artificial intelligent methodologies in different environment such as in telecommunication networks, transportation systems and health care for Personalized Medicine.
The designed and developed approaches in the field of telecommunication networks is presented in Chapter 2, where a multi-agent reinforcement learning algorithm was designed to implement a *model-free* control approach in order to regulate and improve the level of satisfaction of the users, while the research activities in the field of transportation systems are presented at the end of Chapter 2 and in Chapter 3, where two approaches regarding a Reinforcement Learning algorithm and a Deep Learning algorithm were designed and developed to cope with tailored travels and automatic identification of transportation moralities. Finally, the research activities performed in the field of Personalized Medicine have been presented in Chapter 4 where a Deep Learning and Model Predictive control based approach are presented to address the problem of controlling biological factors in diabetic patients.

Le metodologie di Intelligenza Artificiale (AI) si occupano della possibilità di rendere le macchine in grado di compiere azioni intelligenti con lo scopo di aiutare l'essere umano; quindi è possibile affermare che l'Intelligenza Artificiale consente di portare all'interno delle macchine, caratteristiche tipiche considerate come caratteristiche umane.
Nello spazio dell'Intelligenza Artificiale ci sono molti compiti che potrebbero essere richiesti alla macchina come la percezione dell'ambiente, la percezione visiva, decisioni complesse.
La recente evoluzione in questo campo ha prodotto notevoli scoperte, principalmente in sistemi ingegneristici come sistemi multi-agente, sistemi in rete, impianti, sistemi veicolari, sistemi sanitari; infatti una parte dei suddetti sistemi di ingegneria è presente in questa tesi di dottorato.
Lo scopo principale di questo lavoro è presentare le mie recenti attività di ricerca nel campo di sistemi complessi che portano le metodologie di intelligenza artificiale ad essere applicati in diversi ambienti, come nelle reti di telecomunicazione, nei sistemi di trasporto e nei sistemi sanitari per la Medicina Personalizzata.
Gli approcci progettati e sviluppati nel campo delle reti di telecomunicazione sono presentati nel Capitolo 2, dove un algoritmo di Multi Agent Reinforcement Learning è stato progettato per implementare un approccio *model-free* al fine di controllare e aumentare il livello di soddisfazione degli utenti; le attività di ricerca nel campo dei sistemi di trasporto sono presentate alla fine del capitolo 2 e nel capitolo 3, in cui i due approcci riguardanti un algoritmo di Reinforcement Learning e un algoritmo di Deep Learning sono stati progettati e sviluppati per far fronte a soluzioni di viaggio personalizzate e all'identificazione automatica dei mezzi trasporto; le ricerche svolte nel campo della Medicina Personalizzata sono state presentate nel Capitolo 4 dove è stato presentato un approccio basato sul controllo Deep Learning e Model Predictive Control per affrontare il problema del controllo dei fattori biologici nei pazienti diabetici.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Artificial Intelligence (AI) methodologies deal with the possibility to make machines able to perform intelligent actions with the aim of helping the human being; then is possible to assert that Artificial Intelligence permits to bring into the machine, typical characteristics considered as humans. In the AI space there are a lot of tasks that could be demanded to machine such as environment aware perception, visual perception, complex decisions.
AI is experiencing a moment of great interest in the scientific community, as never before, thanks above all to the current computation power of the modern computers. Currently, several different research activities are really active to bring AI practically everywhere, and some industrial companies are ready to install working AI tools into the services and things. Bringing *intelligence* within things means not only have machines/computers with relevant computational power but also endow the machines with reasoning capabilities. The recent evolving in this field has produced remarkable upgrades mainly on engineering systems such as multi-agent systems, networked systems, manufacturing systems, vehicular systems, health care systems, etc... .
The main scope of my recent research activities is a tentative approach to use intelligent control [1] [77], *a data driven control framework* [47] [102], which uses typical artificial intelligence methodologies, such as Neural Network [110], Genetic Algorithms [91] and Reinforcement Learning [56]. The data-driven control appears when traditional control methods can not address real-world systems. The data driven control (often found in the literature relative to data driven modeling) are based on the analysis of data for a generic system and are in charge of finding the relations between the system variables (input, internal and output) without having a prior knowledge of the physical dynamics of

the system [90]; such methods are for instance the three branches of Artificial Intelligence: Machine Learning, Deep Learning and Reinforcement Learning, all of which are able of extracting patterns from high dimensional data discovering models of dynamical systems or learning control laws, directly from the data. In some traditional control methods the decision about the control actions are mainly based on received feedback, for example the sensor measurements, and the system can be set up with a control strategy that corrects and drives the whole system, such a pendulum, to stabilize it and to follow the desired trajectory. The control system for the pendulum, can be explicitly determined since we have the dynamical model, but in some real-world scenario, such as neuroscience we don't have an analytical representation of the model and hence it looks as practically unfeasible applying traditional control strategies. In fact, for systems such as neuroscience, finance, climate and self-driving cars, it is really difficult to identify a model, due to the massively nonlinear behavior. In some worst cases, the model doesn't exist at all, since the equations of such systems are unknown. For such systems where the dynamics is unknown one cannot develop a traditional control law, and hence the strength of data-driven control is clear. In this context, the modern solutions brought from model-free approaches such as Reinforcement Learning, Deep Learning, Machine Learning in general (i.e., modern AI solutions), can address the possibility to control systems where we might have unknown dynamics, or we are facing with nonlinear system very high dimensional states and limited measurements. The machine learning control is that in all of the mentioned complex system with high dimensional nonlinear, possibly unknown, dynamics tend to produce the dominant patterns that emerge from the data. The dominant patterns created and get out with machine learning methodologies from the data can represent the what we should control, creating hence a data-driven model.
In this respect, and according to my PhD supervisor, I have conducted research activities with the scope of developing novel AI solutions; the solutions have been developed and tested in various scenarios such as in telecommunication networks, transportation systems and health systems for Personalized Medicine. The research activities were also conduced, during the PhD period, for an in-house consultant @CRAT (Consorzio per la Ricerca nell'Automatica e nelle Telecomunicazioni) for developing end-to-end algorithms and/or applications. At CRAT I have actively participated (this role is still active) in several different

funded projects from both Italian (MIUR) and European side (H2020).
The projects at issue are:

- PLATINO (MIUR-PON project): Platform for Innovative Services in Future Internet
- BONVOYAGE (H2020 project): From Bilbao to Oslo, intermodal mobility solutions, interfaces and applications for people and goods, supported by an innovative communication network
- 5G ALLSTAR (H2020 project): 5G AgiLe and fLexible integration of SaTellite And cellulaR

The PLATINO project was focused on designing and developing a service platform able to provide heterogeneous services and contents to end users by evaluating their satisfaction during the experienced content. The platform had to be so flexible enough to satisfy at the same time a lot of terminals while guarantying a personalized Quality of Experience (QoE) based on the evaluation of the perceived Quality of Service (Qos).
The problem of QoS and related QoE in telecommunication networks is widely investigated in the literature [13], [12], [86], [88], [41] and is aimed at controlling the network status even when the resources are not enough for all the connected terminals. In PLATINO project my role was to investigate, design and implement a realistic framework application for evaluating and controlling the QoE trying to guarantee the highest possible QoS to all the end users also in presence of a huge number of connected terminals.
To address this problem I have implemented a Multi Agent Reinforcement Learning (MARL) control algorithm based on an innovative heuristic solution implemented for finding a consensus among users. The heuristic solution, MARL based, was able to dynamically provide the most appropriate Class of Service (CoS) [104] to the connected terminals; the CoS was computed according to the status of the overall network and also considering the users' needs in terms of QoS. A personalized parameter, that is QoE, was then computed as the expression of users' personalized perception for the on-going contents (e.g., Movie Streaming, Conference Call, etc..). The CoS, that was associated dynamically to each user and content, was aimed at incrementing the overall performances in order to guarantee the Personalized QoE needs.
The research activities and the achieved results following the mentioned approaches in this field have led to two publications:

- F.Delli Priscoli, A. Di Giorgio,F. Lisi, S. Monaco, A. Pietrabissa, L. Ricciardi Celsi,and V. Suraci, "*Multi-agent quality of experience control*", International Journal of Control, Automation and Systems,Volume 15 number 2, pages 892-904, 2017.

- S. Battilotti, S. Canale, F. Delli Priscoli, L .Fogliati, C. Cori Giorgi, F. Lisi, S. Monaco, L. Ricciardi Celsi and V.Suraci, "*A Dynamic Approach to Quality of Experience Control in Cognitive Future Internet Networks*", poster appearing in Proceedings of the 24th European Conference on Networks and Communication (EuCNC 2015)

The BONOVOYAGE project deals with the possibility to design, develop, and implement a platform able to multi-modal door to door scalable solutions for passengers and goods. The main impact of the project was to realize an innovative communication network for sharing National and International transport information in order to create a set of National Access Point where the whole set of transportation provider information was stored. The really challenge was to improve the current Transportation Systems bringing an innovative way to manage and distribute transportation providers information and bring intelligence in journey planner solutions. The whole platform was assisted and make real by an APP where users are able to interact for looking for innovative trip solutions and providing feedbacks in terms of expected quality of the overall system. The system is able to acquire information from users' feedbacks and use them to improve itself.
My role in this project was to design and implement efficient algorithms able to acquire knowledge from users' expectations based on their feedbacks and choices during their interaction with APP. Such approaches was devoted to provide sophisticated and personalized travel solution for each user while using the developed BONVOYAGE APP.
During the BONVOYAGE projects new research activities was conducted in the field of transportation system, where innovative algorithms were designed and developed to identify almost in real-time the transportation mode with which users are moving. Such an approach could promote, at least in my vision, the development of traffic control and management algorithms.
The results of such works have led to the following papers/submitted papers:

- S. Canale, A. Di Giorgio, F. Lisi, M. Panfili, L. Ricciardi Celsi, V. Suraci, and F. Delli Priscoli, "*A Future Internet Oriented User Centric*

Extended Intelligent Transportation System," in Proceedings of the 24th Mediterranean Conference on Control and Automation (MED 2016),pp. 1133-1139, June 21-24, 2016, Athens, Greece

- A. Detti, G. Tropea, N.Blefari Melazzi, D. Kjenstad, L. Bach, I. Christiansen, F. Lisi, "*Federation and Orchestration: a scalable solution for EU multimodal travel information services*", submitted to Intelligent Transportation System Magazine, 2018

- A. Giuseppi, F. Lisi, A. Pietrabissa, "*Automatic Transportation Mode Recognition on Smartphone Data based on Deep Neural Networks*," submitted to Journal of Intelligent Transportation Systems: Technology, Planning, and Operations, 2018

The ongoing 5G-ALLSTAR project deals with the possibility to manage multi-connectivity based services, in order to create a sort of cooperation among different Radio Access Technologies. The cooperation begins with the sharing of the spectrum among different radio access and supports the great challenges to use multiple Radio Access technologies at the same time for the same transmission by joining the resources of the Cellular (i.e., 5G NR, 4G LTE, 3G) and Satellite networks.
My role in this project, just started, will be to identify potential methodologies to fight with the actual multi-connectivity challenge that is aimed at introducing advanced algorithms able to provide appropriate and scalable algorithms for traffic steering problems. The solution, which includes the traffic steering algorithms, should also take into account the users' expectations.
The research activities of the 5G-ALLSTAR project are out of scope of this thesis and not will be reported.

It is clear that the Artificial Intelligence methodologies and algorithms have been the basis of my involvement in the PLATINO, BONVOYAGE and 5G ALLSTAR projects. In fact, even if in different scenarios, similar methodologies were applied by modeling them in different environments.
The research activities were not conducted only within the above mentioned European projects, and the techniques acquired were used, notably, in a new research argument concerning Prediction and Control for biological factors.
The common ground with the other research activities presented in this chapter is the use of Artificial Intelligence techniques in the field of health for Per-

sonalized Medicine. In this respect, a Deep Learning and Model Predictive Control based innovative solution has been designed and tested, by using an in-silico Application able to simulate patients afflicted by diabetes, to control and maintain the blood glucose level in what is commonly considered as the safe range. The proposed solution concerns the usage of Deep Bidirectional Neural Network for predicting the blood glucose level (glycemic index) in sick patients and controlling that index with a Model Predictive Control technique able to determine instant by instant the more appropriate insulin injection to patient itself. The effectiveness of the proposed solution was tested on an in-silico patients towards an application developed by UVA/PADOVA [71]. The mentioned solution will lead to a international publication in a journal paper, that is not yet finalized and in this thesis a draft version is presented.

The PhD thesis is structured as follow:

- Chapter 2: State of the Art and Multi-Agent Reinforcement Learning algorithms applied in both Telecommunication networks and vehicular networks
- Chapter 3: State of the Art and Deep Learning based Transport recognition system
- Chapter 4: State of the Art and Deep Model Predictive Control based diabetes application

In Chapter 2 a complete description of the Multi Agent Reinforcement Learning state of the art is presented as well as its application and relative obtained results in the PLATINO project and BONVOYAGE project. In Chapter 3 after a brief introduction about Machine Learning and Deep Learning architectures and algorithms, a Transportation Mode Recognition application is presented also discussing the main achievements. In Chapter 4 the state of the art about recent advances in the filed of biological factors such as glucose level control in patients afflicted by diabetes is presented and an innovative Deep Learning and Model Predictive Control based, namely Deep Model Predictive Control, solution is proposed in its draft version; it is indeed in a draft version but already deserved to be included in this work.

Chapter 2

Reinforcement Learning based Multi Agent Control Systems

In this Chapter I report the research activities carried out in the field of *Reinforcement Learning*, *Game Theory* and *Multi-Agent Reinforcement Learning*. More precisely, the methodologies introduced in the first part of this chapter (see Section 2.1 and 2.2) have been extended in order to be applied in several use cases. Section 2.3 presents the research activities performed to cover issues in telecommunication networks where multiple users are involved in sharing bandwidth by exploiting and extending *Multi Agent Reinforcement Learning* approach. The telecommunication networks have to be able to satisfy all the users expectation, assuming bandwidth with limited capacities, and the objective is to assure suitable performances. Section 2.3 shows the results of the publication [22].
Section 2.4 presents the research activities conducted in the framework of BONVOYAGE [1] H2020 project for addressing the possibility to introduce artificial intelligence techniques in the Intelligent Transportation System [25], [54], and [100]. More precisely, Section 2.4 reports: (i) a publication [17] in the field of Machine Learning methodologies to identify travellers' profiles for computing tailored trips and (ii) a *Reinforcement Learning* algorithm properly designed and implemented to rank tailored journey solutions according to a new deducted travellers' model.

[1] BONVOYAGE H2020 http://www.bonvoyage2020.eu/

2.1 Reinforcement Learning

Reinforcement Learning is an area of autonomous learning that deals with the possibility to create an agent able to perform actions or make decisions in an unknown environment. At the beginning the agent needs to perform several actions within environment to be able to acquire the proper knowledge. The environment is modeled as a set of states; each agent can perform actions according to the belonging state, receiving a suitable reward for each action performed in each state.
The main goal in Reinforcement Learning is to achieve the final state by acquiring the highest possible cumulative reward.
The Reinforcement Learning approach can be used in several different domains [55]; I have investigated two domains for solving problems using Reinforcement Learning. In particular, the domains in question are game theory and control technique.
In Reinforcement Learning problems the environment is typically formulated as a Markov Decision Process (MDP) [51]. An MDP can be modeled as a tuple:

$$(S, A, P_a, R_a, \gamma) \tag{2.1}$$

S: represents the state space,
A: represents the actions space,
P_a: represents the probability to choose an action in a given state,
R_a: represents the reward for each performed action in a given state
γ: represents the discount factor assumed to be in the range of $[0, 1]$.
The reward, received by an agent, can be considered as the feedback received by the agent while performing an action in its current state at time t.
The selection of the best action is called policy and an agent can learn its best policy, selecting each time the action that produce the greatest possible cumulative reward.
The selection of the best action in each state is the definition of policy. The agent can learn its best policy, by selecting during the interaction with the environment the actions that produce the greatest possible cumulative reward.
Considering a state-value function for a policy π [94]:

$$V^{\pi}(s) = E_{\pi}\{R_t|s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\} \tag{2.2}$$

In equation (2.2) is computed the expected return for an agent when it start in state s_t, and when the agent follows the policy π, the $E_{\pi}\{\cdot\}$ is the expected value. The discounted factor is defined by γ.
However, the value of an action, for an agent, taking action a in state s and when it follows the policy π, is represented by:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t|s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s, a_t = a\} \tag{2.3}$$

The optimal expected return for the state-value function can be obtained in a MPD using the Bellman equation [10] for V^{π}.
The method used in RL to estimate the optimal value of a state is :

$$V^{\pi}(s) = \sum_{a}\pi(s,a)\sum_{s'}P^a_{ss'}[R^a_{ss'} + \gamma V^{\pi}(s')] \tag{2.4}$$

and the more visits has a state the better estimation of $V^{\pi}(s)$ is obtained.
Anyway the aim of an agent is try to maximize its reward over time, this means getting an optimal policy. A policy π is better than another in any state s if $V^{\pi}(s) \geq V^{\pi'}(s)$. The Equations (2.2) and (2.3) can be optimized as:

$$V^*(s) = max_{\pi}V^{\pi}(s) \tag{2.5}$$

$$Q^*(s,a) = max_{\pi}Q^{\pi}(s,a) \tag{2.6}$$

The equations (2.5) and (2.6) can be used to compute the optimal value of actions in a state $Q^*(s,a)$ and optimal value of a state $V^*(s)$, when the agent uses the policy π. If the transition probability $P_{ss'}$ and the reward function $R^a_{ss'}$ with the Bellman Equation for $V^*(s)$ can be computed the value of a state s,

$$V^*(s) = max_a\sum_{s'}P^a_{ss'}[R^a_{ss'} + \gamma V^*(s')] \tag{2.7}$$

and the (2.7) using the Bellman Equation, became

$$Q^*(s,a) = \sum_{s'}P^a_{ss'}[R^a_{ss'} + \gamma max_{a'}Q^*(s',a')] \tag{2.8}$$

The Bellman Equation is the basis for many dynamic programming methods to solve MDP, but only when the agent has a perfect model of the environment, knowing the transition probability and rewards function for every action performed in every state.
Some MDP can be solved by using several different techniques such as dynamic programming in case the agent has a perfect knowledge of the environment. This means that the agent knows the P_a (transition probability functions) and the R_a (the rewards) for each state and then it achieves the best possible cumulative rewards.
The environment in which the agent plays can be completely unknown by the agent itself.
This is the case in which the agent can perform actions by exploring the environment. In order to solve a Reinforcement Learning, if the agent has no knowledge of the environment, a **Q-Learning** algorithm can be used.
The Q-Learning [105] Algorithm allows agents to act in an optimal way in Markovian domains, learning by the consequences of action without having knowledge about the transition and reward functions, this is the reason why Q-Learning is a form of model-free Reinforcement Learning.
The aim of Q-Learning, that can be viewed as temporal-difference (TD) learning method, is to estimate the Q-value for each policy. The agent uses its experience during the learning process to improve its estimate. Once the agent has acquired a perfect knowledge of the transition and immediate reward function it can provide the *optimal policy*, and using the equation (2.7) it can calculate the optimal action a_t for each state s_t. The fundamental requirement is that the agent must visit each state often enough to converge (as illustrated in [105]) to the optimal Q^*.
Hence, the sequence of visits of each state permits to the agent to achieve the suitable experience; each visit is performed in a distinct episode.
The Q-Learning Algorithm:

a. Observe s_t to be the current state

b. Choose an action a_t

c. Observe s'_t to be the next state and receive an immediate r_r reward

d. Update the Q_t values, with the form :

$$Q_t(s_t, a_t) = (1 - \alpha_t)Q_{t-1}(s_t, a_t) + \alpha_t[r_t + \gamma max_{a'}Q_{t-1}(s'_t, a'_t)] \qquad (2.9)$$

with γ the discount factor and α the learning rate.
The equation (2.9) takes into account the value of each action in each state, and it updates its estimate of the optimal value of each action in all states. The Q-Learning build a table, called Q-Table, that contains the values, updated with the equation (2.9), of each action-state pair. The learning rate is an important parameter since decreases with the time, and different learning rate might be used for each state-action pair.

2.2 Multi Agent Reinforcement Learning

The Reinforcement Learning algorithms are designed for one agent interacting with its environment. However, Reinforcement Learning problems can be extended for more than one agent interacting with the same environment for multi-agent domains.
As introduced in the previous chapter, the single agent Reinforcement Learning problems can be modeled with Markov Decision Proces; in case of multi agent scenarios, the Multi Agent Reinforcement Learning can be modeled as a game theory problem or stochastic games (SGs) [89].

2.2.1 Matrix Games

The Matrix Games [67] consist in a many players games, e.g. two players games. The players in a game might have opposite or different rewards. The first case is the zero-sum game, the second case is a general-sum game. In a zero-sum game in case of scenario with two agents and the first receives 1 as reward, the second player receives -1 as reward. In a general-sum game each player receives a different reward. In both cases the rewards depend on the actions, chosen from the available actions in action space, of all players. The Matrix game can be formalized by a tuple:

$$< n, A_1, A_2, ..., A_n, R_1, R_2, ..., R_n >$$

where the number of players n have a finite action space A_i and reward R_i, with $i = 1, .., n$.
The rewards for each player are included in a matrix, stored depending on their actions. In case of two-player game, one has two $m \times n$ matrices, X and

Y. In both matrices, Player A is the column player and Player B is the row player, the values in the matrices are represented by the rewards received by the players, depending on the actions chosen.
The Matrix X contains the rewards for Player A, obtained with the joint actions of the Player B, and Matrix Y contains the rewards for Player B, obtained with the joint actions of the Player A, e.g., the position x_{ij} (associated to X) describe the rewards obtained by Player A, when it performs actions i with Player B that performs action j, the y_{ij} (associated to Y) describe the rewards obtained by Player B, when it performs action j with Player A that performs action i. The expected payoff for both players can be represented as :

$$E\{R_X\} = \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i^X x_{ij} \pi_j^Y \tag{2.10}$$

$$E\{R_Y\} = \sum_{i=1}^{m}\sum_{j=1}^{n} \pi_i^X y_{ij} \pi_j^Y \tag{2.11}$$

The policy π_i^X in the equations (2.11) and (2.12) is the probability that Player A will choose action i and π_j^Y is the probability that Player B will choose action j.
A typical example to understand the concept may be "The Matching Pennies Game". In this game with two players, and where each player has one penny, they must show one side of their pennies. When both pennies, of the two players, have the same side, Player A wins and receives a reward 1, while Player B loses and receives reward -1, otherwise if the sides of the pennies are different Player B wins and receives the reward 1, while Player A receives reward -1. This is an example of zero-sum game, since the rewards received by the players are opposites.
The Matrices X=-Y, containing the rewards previously assigned, and can be represented as follows:

$$X = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, Y = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

Table 2.1 Reward Matrices for two players

The rows and the columns of each matrix represent the available actions (show heads or show tails) for row Player A and for column Player B. Each player in this game, has to make a decision each turn. If both players have the

same probability to choose an action, they receive the same amount of rewards, hence the solution, for the players, could be to play one side for half of choices, and the other side for the other half of choices. In this way the total amount of the rewards will be maximized for both Players, and an equilibrium will be reached. The reached equilibrium is called Nash Equilibrium, as properly defined in the following section.

2.2.2 Nash Equilibrium

The Nash Equilibrium, as defined in [89], "*is a collection of strategies for each of the players such that each players' strategy is a best response to the other players' strategies; at a Nash equilibrium, no player can do better by changing strategies unilaterally given that the other players don't change their Nash strategies. There is at least one Nash equilibrium in a game*".
The collection of all players' strategy $(\pi_1^*, \pi_2^*, ..., \pi_n^*)$ in a matrix game is the Nash Equilibrium if:

$$V_i(\pi_1^*, ..., \pi_i^*, ..., \pi_n^*) \geq V_i(\pi_1^*, ..., \pi_i^*, ..., \pi_n^*), \forall \pi_i \in \Pi_i, i = 1, ..., n \tag{2.12}$$

where V_i is the i's value function of the expected reward with all players' strategies, and π_i is any strategy of player i from Π_i strategy space. In [30] there is the proof that in a n-player game there exist at least one mixed strategy equilibrium. In the Nash Equilibrium the players' strategy is influenced by the strategy of all players who try to maximize their rewards, if all strategies are the best the Nash equilibrium is reached.
For two players, in order to find the equations to maximize the expected value, the Nash Equilibrium can be used, and a general example is then explained. As in earlier section for the two player two matrices can be defined:

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}, Y = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$$

Table 2.2 Reward Matrices for two players

In order to maximize V_1 for Player A and V_2 for Player B, to find the optimal π_1^X, π_2^X, π_1^Y and π_2^Y. Now, getting the inequalities:

$$x_{11}\pi_1^X + x_{12}\pi_2^X \geq V_1 \tag{2.13}$$

$$x_{21}\pi_1^X + x_{22}\pi_2^X \geq V_1 \tag{2.14}$$

$$\pi_1^X + \pi_2^X = 1 \tag{2.15}$$

$$\pi_i^X \geq 0, i = 1, 2 \tag{2.16}$$

$$y_{11}\pi_1^Y + y_{12}\pi_2^Y \geq V_2 \tag{2.17}$$

$$y_{21}\pi_1^Y + y_{22}\pi_2^Y \geq V_2 \tag{2.18}$$

$$\pi_1^Y + \pi_2^Y = 1 \tag{2.19}$$

$$\pi_i^Y \geq 0, i = 1, 2 \tag{2.20}$$

Using the linear programming for the above equations (from 2.16 to 2.21) can be found the values for π_1^X, π_2^X, π_1^Y and π_2^Y. Getting back to the example of matching pennies, where there are two players, the associated rewards matrices and the equations (considering $\pi_2^X = 1 - \pi_1^X$) are (for Player A):

$$\begin{array}{c} 2\pi_1^X - 1 \geq V_1 \\ -2\pi_1^X + 1 \geq V_1 \\ 0 \leq \pi_1^X \leq 1 \end{array} \tag{2.21}$$

The optimal value of 0.5 for π_1^X (i.e. Player A) and 0.5 for π_1^Y (i.e. Player B) with the same procedure, using the linear programming. Hence, there is the proof of previous deduction that the best strategy for this game is play one side for half of actions, and the other side for the other actions.

2.2.3 Stochastic Game

The stochastic games [14] - [107] are the generalization of the Markov Decision Process for the multi agent case, and then the fundamental Markov Assumption is still needed. When multiple agents interact in the same environment, the

state transitions and the rewards depend on the joint action of all agents.
The stochastic game can be formalized by a tuple:

$$< S, A_1, ..., A_n, R_1, ..., R_n, T >$$

where S is the space state, A_i is the set of action of agent i, $R_i = S \times A_1 ... \times A_n$ is the payoff function for player i and $T : S \times A^1 ... \times A^n \times S \longrightarrow [0, 1]$ is the transition function.
The transition function is the probability distribution over next state given the current state and the joint action of the agents.
If all agents have the same goal, this mean that all payoffs are equal (i.e. $R_1 = \cdots = R_n$), and the stochastic game is fully cooperative. With only two agents, a 2-player stochastic game can be considered as fully competitive, when the rewards of the agents are opposite. Stochastic Games are mixed games when are neither fully competitive nor fully cooperative.

2.2.4 Nash Q-Learning

The Nash Q-Learning Algorithm [48] - [49] is suited for multi-agent general-sum stochastic game and it is founded on the Q-Learning algorithm for the single-agent case, adapted for multi-agent environment. It is based on the Nash Equilibrium where "*each player effectively holds a correct expectation about the other players' behaviors, and acts rationally with respect to this expectation*" [49]. The strategy of an agent is the strategy that is the best response for all other agents' strategies, and when the strategy change this means that the agent may be in a worst position.
Now, by considering the Q-Learning Algorithm where the agent objective is to maximize its own payoff, building a Q-Table, and adapting Q-Learning to multi-agent general-sum stochastic game, the agents' payoff is given by the joint action of all other agents. The agent needs to take into account the other agents' actions and rewards, collected in the Q-Tables (a Q-Table for each agent) and the Nash Q-values. The Nash Q-Value is defined by Hu and Wellman as "*the sum of discounted rewards of taking a joint action at current state (in state s_0) and then following the Nash equilibrium strategies thereafter*" [49]. In this situation, the agents try to reach an equilibrium (Nash equilibrium) in order to maximize its payoff, and then the optimal strategy will be reached by observing all agents' actions and rewards. Keeping the Q-Learning algorithm update method, i.e, equation 2.9 , the difference with the Nash Q-Learning is that its

update method can be obtained by using the future Nash Equilibrium, take into account the actions of all agents, instead of a single maximum payoff. In [49] is defined the Equation for the Nash Q-function as :

$$Q_*^i(s, a^1, ..., a^n) = r^i(s, a^1, ..., a^n) + \gamma \sum_{s' \in S} p(s'|s, a^1, ..., a^n) v^i(s', \pi_*^1, ..., \pi_*^n) \tag{2.22}$$

where $(\pi_*^1, ..., \pi_*^n)$ is the Nash Equilibrium; each π represents all the policies from the next state until the end state, which can be considered as the series of actions that maximizes the agents' rewards , and then the Nash Equilibrium strategy.
The agent's reward in a state s where all agents perform their actions is given by $r^i(s, a^1, ..., a^n)$ with $(a^1, ..., a^n)$, that represents the joint action, $v^i(s', \pi_*^1, ..., \pi_*^n)$ represents the agent's total discounted reward from the state s' when the agents in the game follow the equilibrium strategies.
Figure 2.1 represents the game for two agents. The left agent must reach point B and the right agent, must reach point A.

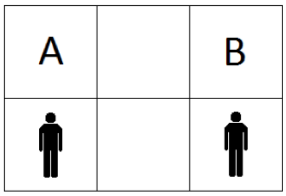

Fig. 2.1 Grid Game

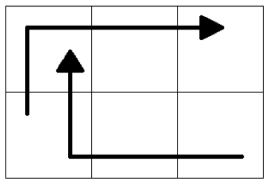

Fig. 2.2 Nash Strategy

Figure 2.2 represent a possible Nash Equilibrium strategy.
The function 3.1 provides a method to update the Q-Tables with the Nash Q-Values, which represent the current reward and future reward for an agent that follows a Nash Equilibrium. For an agent, the Nash Q-Value depends on the joint action strategy and not its own payoff only. This is not always permitted in an environment, in fact there exist several environments in which this is not feasible, but the agents need to identify the other agents' rewards as well. Therefore, the Nash Q-Learning algorithm uses the equation (as defined in [26]) :

$$Q^i_{t+1}(s, a^1, ..., a^n) = (1 - \alpha)Q^i_t(s, a^1, ..., a^n) + \alpha[r^i_t + \gamma NashQ^i_t(s')] \quad (2.23)$$

where,

$$NashQ^i_t(s') = (\pi^1(s'), \pi^2(s'), ..., \pi_n(s'))Q^i_t(s') \quad (2.24)$$

"*Different methods for selecting among multiple Nash equilibria will in general yield different updates.* $NashQ^i_t(s')$ *is agent* i*'s payoff in state* s'*, for the selected equilibrium*" [49].
Given an agent, the Q-Values of the other agents may not be available for it. Thus in order to obtain the Nash Equilibrium the agent must learn these information. Hence, agent i starts to initialize the Q-Function as $Q^j_0(s, a^1, ..., a^n) = 0$ for all other agents (j), and then it observes, during the game, the agents' immediate rewards. The information, obtained during the game are used to update the Q-Function of each agent, previously initialized, through the update rules [49]:

$$Q^j_{t+1}(s, a^1, ..., a^n) = (1 - \alpha)Q^j_t(s, a^1, ..., a^n) + \alpha[r^j_t + \gamma NashQ^j_t(s')] \quad (2.25)$$

Therefore equation in 2.25 "*does not update all the entries in the Q-Functions. It updates only the entry corresponding to the current state and the actions chosen by the agents. Such updating is called asynchronous updating*" [49].

Algorithm 1 Nash Q-Learning Algorithm

Initialization

Let $t = 0$ and initial state s_0

Let each learning agent be indexed by i

For all $s \in S$ and $a^j \in A^j$ with $j = 1, ..., n$

Let $Q^j_t(s, a^1, ..., a^n) = 0$

Loop

Choose $action^i_t$

Observe $r^1_t, ..., r^n_t$ and $s_{t+1} = s'$

Update Q^j_t for $j = 1, ..., n$

$Q^j_{t+1}(s, a^1, ..., a^n) = (1 - \alpha)Q^j_t(s, a^1, ..., a^n) + \alpha[r^j_t + \gamma NashQ^j_t(s')]$

Let $t = t + 1$

Algorithm 1 represents in details the Nash Q-Learning (as described in [49]).
The authors in [49] considers the convergence of the Nash Q-Algorithm under several important assumptions :
Assumption 1. *One of the following conditions holds during learning.*[49]

Condition A. *Every stage game $Q_t^1(s), ..., Q_t^n(s)$, for all t and s, has a global optimal point, and agents' payoff in this equilibrium are used to update their Q-Functions.*[49]

Condition B. *Every stage game $Q_t^1(s), ..., Q_t^n(s)$, for all t and s, has a saddle point, and agents' payoff in this equilibrium are used to update their Q-Functions.*[49]

Assumption 2. *Every state $s \in S$ and action $a^k \in A^k$ for $k = 1, ..., n$ are visited infinitely often.*[49]
Assumption 3. [26] *The learning rate α_t satisfies the following conditions for all $s, t, a^1, ..., a^n$:*

a. $0 \leqslant \alpha_t(s, a^1, ..., a_n) < 1, \sum_{t=0}^{\infty} \alpha_t(s, a^1, ..., a_n) = \infty$,
$\sum_{t=0}^{\infty} [\alpha_t(s, a^1, ..., a^n)]^2 < \infty$, *and the latter two hold uniformly and with probability 1.*

b. $\alpha_t(s, a^1, ..., a^n) = 0$ *if* $(s, a^1, ..., a^n) \neq (s_t, a^1, ..., a^n)$. *This means that agent will only update the Q-values, for the present state and actions. It does not need to update every value in the Q-tables at every step.*

An important issue of this algorithm, is that there might be more than one equilibrium, in this case, as suggested in [49], the Lemke-Howson Algorithm [63] might be used to find a Nash Equilibrium. The algorithm, converges within the previous assumptions.

2.2.5 Friend-or-Foe Q-Learning

The Friend-or-Foe Q-Learning (FFQ) [67], [59] is derived from the idea that the **Assumptions 1,2,3** (previous section) create a sort of restriction, in order to guarantee the convergence of Nash-Q, since the Assumption 1 contemplates that every stage game needs to have either a global optimal point or a saddle point [12].

Algorithm 2 Friend-or-Foe Q-Learning Algorithm

Initialization

$\forall s \in S, a_1 \in A_1$ and $a_2 \in A_2$

Let $Q(s, a_1, a_2) = 0$

$\forall s \in S$

Let $V(s) = 0$

$\forall s \in S, a_1 \in A_1$

Let $\pi(s, a_1) = \frac{1}{|A_1|}$

Let $\alpha = 1$ **Loop**

In state s

Choose a random action from A_1, with probability ε

If not a random action, choose action a_1 with probability $\pi(s, a_1)$

Learning In state s'

Agent observes the reward R related to action a_1 and opponent's action a_2 in state s

Update Q-Table of player with

$Q(s, a_1, a_2) = (1 - \alpha) \cdot Q(s, a_1, a_2) + \alpha(R + \gamma \cdot V[s'])$

find $\pi(s, a_1)$ and $V(s)$ with

$V(s) = max_{a_1 \in A_1, a_2 \in A_2} Q_1[s, a_1, a_2]$ in case of friends players, or

$V(s) = max_{\pi \in \Pi(A_1)} min_{a_2 \in A_2} \sum_{a_1 \in A1} \pi(a1) Q_1[s, a_1, a_2]$ if players are foes.

$\alpha = \alpha \cdot decay$

End Loop

In order to reduce the restriction from **Assumption 2** the FFQ, is then implemented by [67] where it always converge by changing the update rules depending on the agents' behavior, *friend* or *foe*. The label as friend or foe must be identified by the other agent. The FFQ can be viewed as an adaptation of the Nash Q-Learning, where the main feature that differentiates it from the Nash-Q Algorithm is that each agent take into account its own Q-Table; the FFQ was developed for n-player but in Algorithm 2 , just for clarity, we represent it for 2-player. As can be noted from Algorithm 2, the update rule (2.25) replace the function $NashQ^i(s, Q_1, ..., Q_n)$ with:

$$max_{a_i \in A_i} Q_i[s, a_1, ..., a_n] \tag{2.26}$$

if the players are friends, or if they are foes, with:

$$max_{\pi \in \Pi(A_i)} min_{a_i \in A_i} \sum_{a_i \in A_i} \pi(a_i) Q_i[s, a_1, ..., a_n] \tag{2.27}$$

where n is the number of players. Clearly the update rule (2.26) is the same as the Q-Learning update rule (2.9), but it is adapted by [67] for multi-agent case. As explained by the author in [67], for n-player game, $NashQ^i(s, Q_1, ..., Q_n)$ is:

$$NashQ_i(s, Q_1, ..., Q_n) =$$

$$max_{\pi \in \Pi(X_1 \times ... \times X_k)} min_{y_1, ..., y_l \in Y_1, ... Y_l}$$

$$\sum_{x_1, ... x_k \in X_1 \times \cdots \times X_k} \pi(x_1) \cdots \pi(x_k) Q_i[s, x_1, ..., x_k, y_1, ..., y_l] \tag{2.28}$$

where X_1 through X_k are the actions that are available to the k friends of player i, and the actions Y_1 through Y_l are available to its l foes. When the agents are "friends", they try to maximize the payoff for its friend, while in case of "foes", they try to minimize their payoff.

2.3 Multi Agent Control Systems applied to telecommunication networks

The complete content of this Chapter (i.e., 2.3) is object of the publication [22].

A Key Future Internet [18] target is to allow applications to transparently, efficiently and flexibly exploit the available resources, with the aim of achieving a satisfaction level that meets the personalized users' needs and expectations. Such expectations could be expressed in terms of a properly defined Quality of Experience (QoE) [5].
In this respect, the International Telecommunication Union (ITU-T) defines QoE as the overall acceptability of an application or service, as perceived subjectively by the end-user [50]: this means that QoE could be regarded as a personalized function of plenty of parameters of heterogeneous nature and spanning all layers of the protocol stack (e.g., such parameters can be related to Quality of Service (QoS), security, mobility, contents, services, device characteristics, etc.).
Indeed, a large amount of research is ongoing in the field of QoE Evaluation, i.e., of the identification, on the one hand, of the personalized expected QoE level (Target QoE) for a given user availing her/himself of a given application in a given context (e.g., see [53], [28] for voice and video applications, respectively), and, on the other hand, of the personalized functions for computing the Perceived QoE, including the monitorable Feedback Parameters which could serve as independent variables for these functions (e.g., see [84]). In particular, several works focus on studying the relation between QoE and network QoS parameters (e.g., see [32]).
Another QoE-related key research issue is that of QoE Control.
Once a QoE Evaluator has assessed the personalized expected QoE level (Target QoE) and the personalized currently perceived QoE level (Perceived QoE), a QoE Controller should be in charge of making suitable Control Decisions aimed at reducing, as far as possible, the difference between the personalized Target and Perceived QoE levels. The interested readers are referred to [84] and [16] for an approach to QoE Evaluation that is fully consistent with this part of the work. Without claiming to present a ready-to-use solution, this chapter provides some innovative hints that could ensure an efficient implementation of the QoE Controller.

Will be described how Control Decisions can practically be implemented via the dynamic selection of predefined Classes of Service; explained how such a dynamic selection can be performed in a model-independent way – in the authors' opinion, any control-based approach relying on any Future Internet model is not practically viable due to the sheer unpredictability of the involved variables [19] – thanks to the adoption of a suitable Multi-Agent Reinforcement Learning (MARL) technique, such as the MARL Q-Learning algorithm presented in [67] and [14]; then, will be discussed the limitations of MARL Q-Learning with respect to practical implementation and how these limitations can be overcome by adopting the proposed new heuristic algorithm, hereafter referred to as H-MARL-Q algorithm; finally, some numerical simulations showing the encouraging performance results of the new algorithm are presented with reference to the proof-of-concept scenario which will be introduced.

2.3.1 QoE Controller

The QoE Controller makes its decisions at discrete time instants t_k, hereafter referred to as *time steps*, occurring with a suitable time period T, whose duration depends on the considered environment (including technological processing constraints).
We assume that each in-progress *application instance* is handled by an Agent i and we define the personalized *QoE Error* at time t_k (indicated as $e_i(t_k)$), relevant to Agent i, as:

$$e_i(t_k) = PQoE_i(t_k) - TQoE_i \tag{2.29}$$

where $PQoE_i(t_k)$ represents the Perceived QoE, i.e., the QoE currently perceived at time t_k by Agent i, and $TQoE_i$ represents the Target QoE, i.e., the personalized QoE which would satisfy the personalized Agent i requirements. So, if this QoE Error is positive, the in-progress application is said to be over-performing, since the QoE currently perceived by the Agent is greater than the desired one, whereas, if the QoE Error is negative, the in-progress application is said to be under-performing.
Note that the presence of over-performing Agents might affect the system performance, since they may require an unnecessarily large amount of resources, which could cause, in turn, the under-performance of other Agents. The goal of the QoE Controller is to guarantee, at every time t_k, a non-negative QoE Error for

all Agents i (for $i = 1, \ldots, N$), i.e., to avoid the occurrence of under-performing applications. Furthermore, if it is not possible to guarantee a non-negative QoE Error for all Agents (e.g., due to insufficient network resources), the QoE Controller should reduce, as far as possible, the QoE Errors of the various Agents while guaranteeing fairness among them. Fairness basically consists in making sure that the QoE Errors experienced by the Agents are kept, as far as possible, close to one another.

As shown in Future Internet Architecture presented in [22], both the Perceived and the Target QoE should be computed by a suitable QoE Evaluator based on suitable Feedback Parameters resulting from the real-time monitoring of the network, as well as from direct or indirect feedbacks coming from users and/or applications. For a more detailed description of the way the QoE functionalities are embedded in the Future Internet architecture ,see [84].

In particular, a promising approach [84] is to relate the computation of the Perceived QoE to the application type (e.g. real-time HDTV streaming, distributed videoconferencing, File Transfer Protocol, etc.) of each in-progress application instance.

Let M denote the total number of application types in the considered environment; let $m \in 1, ..., M$ denote a generic application type; let $i(m)$ denote an Agent (i.e., an application instance) belonging to the m-th application type. Then, the Perceived QoE for Agent $i(m)$, denoted with $PQoE_{i(m)}(t_k)$, is computed as follows:

$$PQoE_{i(m)}(t_k) = g_m(\phi_m(t_k)) \tag{2.30}$$

where $\phi_m(t_k)$ represents a suitable set of Feedback Parameters for the m-th application type, computed up to time t_k, and g_m is a suitable function relating, for the m-th application type, the Feedback Parameters $\phi_m(t_k)$ with the Perceived QoE. The Target QoE, denoted with $TQoE_i$, can be derived from a suitable analysis of the available Feedback Parameters (e.g., by using unsupervised machine learning techniques), or it can simply correspond to a reference value which is assigned by the Telco operator, taking into account the commercial profile of the user.

The work proposes a solution in which the distributed Agents associated to the application instances are embedded in properly selected network nodes (e.g., in the mobile user terminals): the Agents are in charge of the monitoring and actuation functionalities whereas the control functionalities are centralized in

the QoE Controller.

In particular, whenever a new application instance is born, the associated Agent i is in charge of evaluating the personalized Target QoE $TQoE_i$ (which remains unchanged for the whole lifetime of the application instance), of computing its own personalized Perceived QoE $PQoE_i(t_k)$ and of communicating the monitored values to the QoE Controller. As a result, at each time t_k, the QoE Controller, based on the received values for $TQoE_i$ and $PQoE_i(t_j)$ up to time t_k ($i = 1, \ldots, N; j = 0, 1, \ldots, k$), has to choose the most appropriate action $a_i(t_k)$ (for $i = 1, \ldots, N$) which the Agent i should enforce at time t_k, i.e., the most appropriate joint action $(a_1(t_k), a_2(t_k), \ldots, a_N(t_k))$ which the N Agents should enforce at time t_k. At each time t_k, the chosen joint action is broadcast to the N Agents: then, the i-th Agent has to enforce the corresponding action $a_i(t_k)$.

Note that the proposed arrangement is based on the presence of a centralized entity (i.e., the QoE Controller), collecting the Agents' observations, which performs the MARL algorithm and broadcasts the resulting Control Decisions to the Agents. Therefore, any direct signal exchange among the Agents is avoided, thus limiting the overall signaling overhead. The QoE Controller outputs, i.e, the joint action chosen by the QoE Controller, may include for each Agent the choice of QoS Reference Values (e.g., the expected priority level, the tolerated transfer delay range, the minimum throughput to be guaranteed, the tolerated packet loss range, the tolerated dropping frequency range, etc.), of Security Reference Values (e.g., the expected encryption level, the expected security level of the routing path computed by introducing appropriate metrics, etc.), and of Content/Service Reference Values (e.g., the expected content/service mix, etc.).

The QoE Controller has to dynamically select, for each in progress application instance, the most appropriate Reference Values which should actually drive, thanks to suitable underlying network procedures (which are outside the scope of this work), the Perceived QoE as close as possible to the Target QoE (for further details, see [14] where the above-mentioned Reference Values are referred to as Driving Parameters).

However, since the control action has a large number of degrees of freedom, the exploration of the solution space may take a large amount of time, thus making the task of the QoE Controller excessively complex.

A simpler (yet less fine-grained) control task arises if the management of the

underlying networks is arranged into Classes of Service (CoS), as described in [31].
Now, we can assume that each CoS is associated with a predefined set of QoS Reference Values. Nevertheless, the proposed approach can be applied even in the case when each CoS is associated with a set of Reference Values that are not necessarily related to QoS issues only, but also, for instance, to Security parameters, and/or to Content/Service characteristics, etc. Let S indicate the total number of CoSs and let $a_i(t_k) \in \{c_1, c_2, \ldots, c_S\}$ indicate the action performed by the i-th Agent (i.e., the CoS chosen by the i-th Agent) at the time instant t_k.
In current telecommunication networks, a static CoS assignment policy is adopted: each application instance is given a CoS for its entire lifetime; the CoS associated to a given application instance should be the one whose QoS Reference Values satisfy "on the average" the application requirements. Nevertheless, it is evident that such a static association does not take into account either personalized application requirements or contingent situations taking place in the telecommunication networks, such as congestion events. So, a static CoS assignment may generally lead to poor performance in terms of the personalized QoE perceived by each user. Hence, considered the dynamic CoS-to-application assignment as the methodological means to accomplish the above-mentioned goals in terms of QoE Error reduction and fairness. This means that, at each time instant t_k, the QoE Controller has to decide, in real time, which is the most appropriate CoS to be associated with each in-progress application instance (e.g., if the Agents are embedded in mobile user terminals, the QoE Controller decisions can be implemented by inserting the selected CoS identifier in the header of the packets transmitted by the terminals). Up to the authors' knowledge, apart from [19], such a dynamic assignment approach has never been investigated so far.

The problem of designing the QoE Controller algorithm.

It should be evident that, in order to solve this problem by means of traditional model-based control techniques, the QoE Controller should know – or at least estimate – the correlation between its decisions (namely, the selected QoE Controller outputs) and the Perceived QoE. However, no model of the very complex plant regulated by the QoE Controller (namely, the one receiving the QoE Controller outputs in input and producing the Perceived QoE as its output) can be assumed, since it depends on plenty of hardly predictable factors

(such as traffic characteristics of the ongoing applications, network topologies, resource management algorithms, QoE Evaluation methods and so on).
In light of the above, the QoE Controller decision strategy must be learned on line by trial and error. This is why we propose that the QoE Controller makes use of a model-free Multi Agent Reinforcement Learning (MARL) algorithm in order to evaluate, at each time step t_k, the joint policy $\pi(a_1(t_k), a_2(t_k), \ldots, a_N(t_k)) = \pi(a_1, a_2, \ldots, a_N)$ which, once enforced by the Agents, tracks the discussed goals in terms of QoE Error.
The proposed MARL algorithm works on the basis of the observation of a joint reward $r(t_{k+1}, a_1(t_k), a_2(t_k), \ldots, aN(t_k)) = r(t_{k+1}, a_1, a_2, \ldots, a_N)$, i.e., of the numerical reward (the same for all the N Agents) which is received by each Agent at time t_{k+1} as a consequence of the enforcement, at time t_k, of the joint policy $\pi(a_1, a_2, \ldots, a_N)$. The MARL algorithm in question is aimed at maximizing the *long-run return* $R(\pi)$, namely at maximizing the expected discounted return:

$$R(\pi) = E_\pi \sum_{i=0}^{n} \gamma^k r(t_{k+1}, a_1, a_2, ..., a_N) \tag{2.31}$$

where $\gamma \in [0, 1)$ is the discount rate, which weights immediate versus delayed rewards, and E_π denotes the expected value under policy π. In order to set up a MARL problem, we have to select the state space, the action spaces and the reward function.

1. We consider a static game, i.e., a game with only a single state: such an assumption, on the one hand, is not limiting in our context, and, on the other hand, greatly reduces the computational complexity which in MARL is exponential in the number of state and action variables.

2. Following the discussion on dynamic CoS assignment, the action set A_i of Agent i coincides with the set of CoSs, i.e., $A_i = c_1, c_2, \ldots, c_S$, $i = 1, \ldots, N$. In other words, action $a_i(t_k)$, performed by Agent i at time t_k, can be equal to either c_1, or c_2, ..., or c_S. The cardinality of the joint action space $A = A_1 \times \ldots \times A_N$ is equal to $|A_1| \times |A_2| \times \ldots \times |A_N| = S_N$.

3. The function expressing the joint reward $r(t_{k+1}, a_1, a_2, \ldots, a_N)$ should be consistent with the discussed goals in terms of QoE Error; in this respect, each candidate joint reward should be a non-increasing function of the N error values $|e_i(t_k)|$ (for $i = 1, \ldots, N$).

In particular, we propose to apply the Multi-Agent Q-Learning algorithm [67] (hereinafter referred to a MARL-Q algorithm) which is proved to converge to an optimal policy $\pi^*(a_1, a_2, \ldots, a_N)$, i.e., to a policy which maximizes the expected discounted *long-run return* $R(\pi)$ [105]. The MARL-Q algorithm relies on the estimation of the optimal action-value function $Q_\pi(s, a_1, a_2, \ldots, a_N)$, defined as the expected return of the system when it starts from state s, takes the joint action $a_1, a_2, \ldots, a_N$, and follows policy π thereafter. In the previously defined centralized context, at each time step t_k, this algorithm (i) evaluates a joint policy $\pi(a_1, a_2, \ldots, a_N)$ – which sums up the behavior of all the N Agents and is initialized arbitrarily – and (ii) improves such a policy by making it ϵ-greedy with respect to the current action-value function [94], thus yielding a better joint policy π' to be evaluated and improved at the next iteration.
In detail, the policy evaluation step (i) is performed by the MARL-Q algorithm by updating the action-value function $Q(t_k, a_1, a_2, \ldots, a_N)$ according to the following update rule:

$$\begin{gathered} Q(t_k, a_1, a_2, \ldots, a_N) = \\ (1 + \alpha(t_k)) \times Q(t_{k-1}, a_1, a_2, \ldots, a_N) + \alpha(t_k) \times [r(t_k, a_1, a_2, \ldots, a_N) \\ +\alpha \times (t_k) + \gamma max Q(t_{k-1}, a_1, a_2, \ldots, a_N)] \end{gathered} \tag{2.32}$$

where $\gamma \in [0, 1)$ is the discount rate and $\alpha(t_k)$ is a sequence of learning rates, which are key parameters that should satisfy the standard stochastic approximation conditions for convergence [52]. The argument t_k denotes the value of the action-value function computed at time t_k, whereas the argument s is omitted since we consider a single state problem.
The policy improvement step (ii) consists in performing, with probability equal to ϵ, a random joint action $(a'_1, a'_2, \ldots., a'_N)$ and, with probability equal to $1-\epsilon$, the following greedy joint action $(a'_1, a'_2, \ldots., a'_N)$:

$$a'_1, a'_2, \ldots., a'_N = argmax Q(t_k, a_1, a_2, \ldots, a_N) \tag{2.33}$$

The parameter $\epsilon \in$ to $(0, 1)$ is the exploration rate. A large value of ϵ guarantees that different policies with respect to the current best one are explored, and thus avoids that the QoE Controller remains stuck in a local minimum (exploration); on the other hand, a small value of ϵ lets the QoE Controller choose the best action based on the current estimate of the action-value function (exploitation). So, at each time step t_k, the centralized QoE Controller based on the Perceived

QoE values $PQoE_i(t_k)$ $(i = 1, \ldots, N)$ transmitted by the Agents at time t_k, and on the knowledge of the Target QoE values $TQoE_i$ $(i = 1, \ldots, N)$ transmitted by the Agents at the time of their birth – performs the following tasks until the optimal action-value function $Q*$ (and the optimal policy π^*) is found:
T1) it updates the action-value function Q according to (2.32);
T2) it determines the joint action $(a'_1, a'_2, \ldots, a'_N)$ in a random way with probability equal to ϵ, and according to (2.33) with probability equal to $1-\epsilon$;
T3) it broadcasts the chosen joint action $(a'_1, a'_2, \ldots, a'_N)$ to all Agents so that Agent i should enforce action a_i';
T4) it computes the corresponding joint reward $r(t_{k+1}, a_1, a_2, \ldots, a_N)$ according to the selected reward function which should include, as independent variables, the Perceived QoE values $PQoE_i(t_k)$ $(i = 1, \ldots, N)$ and the Target QoE values $TQoE_i$ $(i = 1, \ldots, N)$.
The algorithm converges under a generic initial policy. By varying the learning rates, the exploration rate and the discount rate, the convergence speed of the algorithm and the quality of the solution significantly change; the parameters used in the simulations have been tuned by running the simulations several times.

2.3.2 Proposed Heuristic MARL-Q based (H-MARL-Q) algorithm

The analysis of the contents of the previous section offers us the opportunity to discuss the following issues. The main challenge arisen in MARL is the so-called curse of dimensionality [14]: in fact, as Reinforcement Learning algorithms (such as Q-Learning) estimate values for each possible state or state-action pair, the computational complexity of MARL is exponential in the number of state and action variables and, therefore, in the number of Agents; in addition, the Agents' rewards are correlated and then they cannot be maximized independently of one another.
The runtime of the MARL-Q algorithm (i.e., the time the algorithm needs to perform the specific task it has been designed for) directly depends on the cardinality S^N of the joint action space. As a matter of fact, at each time step, the max operator in (2.33) has to consider S^N values; in this respect, it is particularly important to note that, in a Future Internet framework where the QoE Controller should be able to handle even thousands of Agents and dozens

of CoSs, S^N would become a really huge value. For this reason, the task of implementing the dynamic CoS assignment according to the MARL-Q algorithm discussed in the previous section is inherently complex from a computational point of view and, as a result, it is extremely runtime-consuming.
Such a relevant issue claims for a reasonable reduction of the size of the joint action space (and, hence, of the computational effort of the learning algorithm). Moreover, the issue of non stationarity of multi agent learning arises too, since all Agents in the system are simultaneously learning: each Agent is faced with a moving-target learning problem and consequently the best policy changes as the other Agents' policies change. In this respect, the exploration strategy is crucial for the efficiency of MARL algorithms.
Agents explore to obtain information not only about the environment, but also about the other Agents, for the purpose of implicitly building models of these Agents. In other words, the need for coordination stems from the fact that the effect of any Agent's action on the environment depends also on the actions taken by the other Agents. Nonetheless, too much exploration should be avoided, as it may destabilize the learning dynamics of the other Agents.
In order to address the above-mentioned limitations, this work develops an innovative heuristic algorithm, hereafter referred to as H-MARL-Q algorithm and derived from the MARL-Q algorithm. Such a heuristic algorithm, in comparison with the latter, considerably reduces the joint action space, thus significantly accelerating the task of dynamic CoS mapping, without teasing out an excessive amount of exploratory and information-gathering actions (hence, preserving an acceptable level of environment exploration).
The proposed H-MARL-Q algorithm has also turned out to be successful in addressing the issue of the algorithm scalability, yielding satisfactory results even when the number of Agents is counted in the order of thousands (as it will happen in the upcoming Internet of Things era).
The H-MARL-Q algorithm only considers a suitably selected subset of the joint action space, reasonably yielding an approximate solution to the dynamic CoS assignment problem, as already discussed. Basically, at each time step, the entire joint action space contains plenty of joint actions which have very few possibilities of being the best ones (i.e., the ones which meet the max operator in (2.33)). Unfortunately, such joint actions cannot be identified and discarded a-priori, because we do not have any a-priori knowledge of the environment; nevertheless, such actions can be identified and removed by carrying out a

preliminary analysis of the Agents' dynamic behavior in a simpler emulated environment.
So, the basic underlying idea of the H-MARL-Q algorithm is to perform the following two steps:

- **Step (a)**:
This step, referred to as Identification of the Reduced Joint Action Space, is performed by the QoE Controller una tantum, every time a new Agent is born, in order to identify, through the emulation of suitable test environments, an appropriate Reduced Joint Action Space.

- **Step (b)**:
This step, referred to as Identification of the Suboptimal Joint Action, is performed, in real time, by the QoE Controller at each time step t_k, in order to identify the joint action $(a_1, a_2, \ldots, a_N)$ to be performed at time t_k on the basis of real-time observations of the environment and considering the Reduced Joint Action Space identified in step (a) (and not the entire joint action space A). This yields a suboptimal joint policy which constitutes a satisfactory approximate solution to the considered problem.

Whenever a new Agent, say agent N, is born (i.e., a new application instance is launched), say at time t_k, in a real environment in which N - 1 Agents i (for $i = 1, 2, \ldots, N$ - 1) are already active, the new Agent notifies its existence to the QoE Controller together with its own personalized QoE requirements expressed in terms of Target QoE ($TQoE_N$). Then, the QoE Controller emulates the dynamic behavior of the system in $N - 1$ two-player test games, each one involving two Agents: (i) the new Agent N and (ii) each of the already active Agents i ($i = 1, \ldots, N - 1$). These two-player test games are played in emulated test environments which should reproduce only some key features of the real environment.
Let $[i, j]$ denote the two-player test game involving Agents i and j.
In each two-player test game $[i, j]$ the optimal policy $\pi^*(a_i, a_j)$ is obtained by applying the MARL-Q algorithm described in the previous section (clearly, in this case, the number of Agents N appearing in (2.32) and (2-33) is equal to two). The optimal policy identifies a pair of deterministic actions (a_i^*, a_j^*) where a_i^* and a_j^* represent the optimal CoS choices that the Agents i and j,

respectively, should enforce.
It should be clear that, since the cardinality of the joint action space of each test environment is equal to S^2, the computational complexity of the MARL-Q algorithm is limited, i.e., the algorithm converges to the optimal policy in a limited runtime.
After **step (a)**, at any time t_k at which N Agents are active, the QoE Controller stores $N(N$ - $1)/2$ optimal action couples:

$$(a_i^*, a_j^*) \quad with \quad i = 1, \ldots, N, j = 1, \ldots, N, i \neq j. \tag{2.34}$$

These couples are used in order to identify a Reduced Joint Action Space containing a reasonable subset of the entire joint action space A. Let $a_i^*\ [i,j]$ and $a_j^*\ [i,j]$ denote the optimal action for the i-th Agent and the j-th Agent, respectively, resulting from the two-player test game $[i,j]$. We assume that such a Reduced Joint Action Space consists of the union of N Action Subspaces, where the i-th Action Subspace is associated to the i-th Agent (the sub-tables within the borders in bold in the table below represent such Action Subspaces). Each Action Subspace includes S candidate joint actions (i.e., the rows of each sub-table). The i-th Action Subspace is built by only considering the two-player test games involving the i-th Agent. In particular, each of the S candidate joint actions of the i-th Action Subspace is obtained as follows: for each Agent j, with $j \neq i$, the optimal action $a_j^*\ [i,j]$ that such an Agent would perform in the two-player test game $[i,j]$ is taken into account, whilst for the i-th Agent all the S possible actions of the A_i set are spanned (each one being considered in a different candidate joint action of the Action Subspace). By so doing, the Reduced Joint Action Space includes $S\times$ candidate joint actions: this certainly entails a drastic reduction with respect to the S^N joint actions that would appear in the entire joint action space A. For instance, if, at the considered time step, $N = 4$ (i.e., the Agents 1, 2, 3 and 4 are active) and $S = 3$ (i.e., the action a_i that Agent i (for $i = 1, 2, 3, 4$) can perform corresponds to the selection of one of the three CoSs c_1, c_2, c_3), each of the $S \times N = 12$ lines of the table below provides one of the 12 candidate joint actions (in particular, the sub-tables included within the borders in bold identify the $N = 4$ Action Subspaces), while each of the four columns of the table identifies the single actions that can be taken by Agents $\{1, 2, 3, 4\}$, respectively, in the overall Reduced Joint Action Space.
Moreover, every time a new Agent, say agent N, dies (i.e., an in-progress appli-

Table 2.1 Representation of the reduced joint action space for N=4 and S=3

c1	$a_2^*[1,2]$	$a_3^*[1,3]$	$a_4^*[1,4]$
c2	$a_2^*[1,2]$	$a_3^*[1,3]$	$a_4^*[1,4]$
c3	$a_2^*[1,2]$	$a_3^*[1,3]$	$a_4^*[1,4]$
$a_1^*[1,2]$	c1	$a_3^*[2,3]$	$a_4^*[2,4]$
$a_1^*[1,2]$	c2	$a_3^*[2,3]$	$a_4^*[2,4]$
$a_1^*[1,2]$	c3	$a_3^*[2,3]$	$a_4^*[2,4]$
$a_1^*[1,3]$	$a_2^*[2,3]$	c1	$a_4^*[3,4]$
$a_1^*[1,3]$	$a_2^*[2,3]$	c2	$a_4^*[3,4]$
$a_1^*[1,3]$	$a_2^*[2,3]$	c2	$a_4^*[3,4]$
$a_1^*[1,4]$	$a_2^*[2,4]$	$a_3^*[3,4]$	c1
$a_1^*[1,4]$	$a_2^*[2,4]$	$a_3^*[3,4]$	c2
$a_1^*[1,4]$	$a_2^*[2,4]$	$a_3^*[3,4]$	c3

cation terminates), the Reduced Joint Action Space is updated by eliminating the actions involving Agent N. For instance, referring to the example reported in the table below, if Agent 4 dies, the three joint actions corresponding to the three last rows are removed (i.e., the Action Subspace corresponding to Agent 4 is removed), and all the actions corresponding to the last column are removed, too.

Step (b) of the H-MARL-Q algorithm is performed on the basis of the MARL-Q algorithm and is applied to the Reduced Joint Action Space identified in step (a). So, in step (b), the QoE Controller has to perform the tasks T1, T2, T3, and T4, previously described, with the fundamental difference that, when performing tasks T1 and T2, the Reduced Joint Action Space (having cardinality $S \times N$), instead of the entire Joint Action Space (having cardinality S^N), is considered. Since N can be in the order of thousands, it is evident that the proposed approach drastically reduces the required computing power.

2.3.3 H-MARL-Q algorithm simulation

This paragraph presents numerical simulations, carried out using MATLAB®, with reference to a simple simulation scenario which does not claim to represent any real network. The presented simulations are just aimed at providing a proof-of- concept of the proposed algorithm in order to highlight its potentialities and criticality. We assume the presence of $S = 3$ different CoSs (e.g., "guaranteed,"

"premium" and "best effort" services) and $M = 3$ different application types (i.e., real-time HDTV streaming, distributed videoconferencing and simple File Transfer Protocol).

The static CoS assignment policy determines a static association among application types and CoSs (i.e., an application instance belonging to a given application type is assigned the corresponding CoS for its entire lifetime), whereas in the dynamic CoS assignment case, at each time step t_k, an application instance can be assigned any CoS (regardless of the application type) according to the proposed H-MARL-Q algorithm. We assume that during our simulations N Agents are active, each one being involved in an application instance.

Such an application instance may belong to one of the three considered application types and is characterized by an average offered transmission bit rate b_i randomly selected in the set $\{0.6, 1.2, 2\}$ and by a personalized Target QoE $TQoE_i$ (for $i = 1, \ldots, N$) randomly selected in the set $\{0.7, 0.8, 0.9\}$.

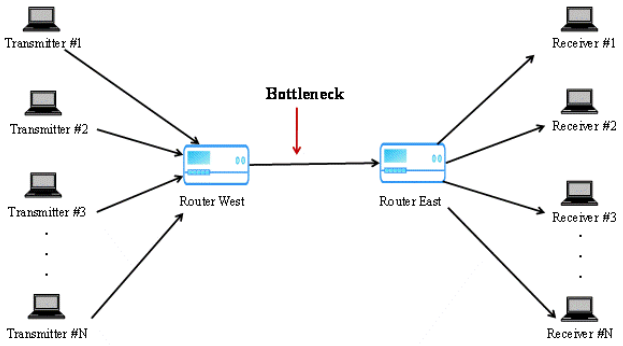

Fig. 2.3 Example Network

The simulated network has a dumbbell network topology, as shown in Fig. 2.3, where each of the N transmitters corresponds to one of the N considered Agents. Router West implements a Weighted Fair Queuing (WFQ) scheduler for handling the traffic to be transmitted over the bottleneck link. The related WFQ vector [23] is assumed to be $(0.2, 0.3, 0.5)$, where the i-th element is the weight assigned to the i-th CoS (higher weight means higher priority). The bottleneck link is characterized by an available link capacity B_{link} computed as:

$$B_{link} = \omega \sum_{i=1}^{N} b_i \tag{2.35}$$

where ω is a parameter in the range $(0, 1)$ accounting for traffic congestion; in particular, in our simulations we consider two different situations characterized by $\omega = 0.7$ and $\omega = 0.8$, which represent High Traffic and Medium Traffic conditions, respectively. As for the number of active Agents N, in our simulations we consider two cases: $N = 100$ and $N = 1000$. For each of these two cases and for each of the two considered traffic congestion conditions, ten simulation runs or episodes have been carried out, with a duration of (15×103) time steps for $N = 100$ and of (15×104) time steps for $N = 1000$: in each simulation run a different association among application instances, application types, average offered bit rates and Target QoE values is performed.
Such associations are assumed to be fixed for the entire simulation run. In the simple proposed simulation scenario, we assume that the set of Feedback Parameters ϕ_m includes, for any $m = 1, 2, 3$, just a single element denoted as ϕ_{QoS} and that the function g_m, is computed on the basis of the well-known IQX hypothesis [32]. This means that (2.30) becomes:

$$PQoE_i(m)(t_k) = p_m e^{\sigma_m} \phi_{QoS} + \tau_m \tag{2.36}$$

where the parameter ϕ_{QoS} has been assumed to be equal to the difference between the traffic offered by the application instance and the corresponding bit rate currently allocated by the WFQ Scheduler.
Note that the latter parameter depends on the CoS appointed at time t_k for the considered application instance, which actually impacts on the priority assigned by the WFQ Scheduler to the packets of the relevant traffic flow. We assume $\sigma 1 = 0.5$, $\sigma 2 = 0.7$, $\sigma 3 = 1$, as well as $p_m = 1$ and $\tau_m = 0$ for all values of m; with these choices, $PQoE_i(m)(t_k)$ is always included in the range $[0, 1]$. The learning rates $\alpha(t_k)$ appearing in (2.32), according to [75], are set to:

$$\alpha(t_k, a_1, a_2, \ldots, a_N) = 1/(1 + visit(t_k, a_1, a_2, \ldots, a_N)) \tag{2.37}$$

where $visit(t_k, a_1, a_2, \ldots, a_N)$ is the number of times that a specific joint action $(a_1, a_2, \ldots, a_N)$ has been enforced up to the iteration at time t_k. The discount rate is set to $\gamma = 0.9$. The selected joint reward function, consistent with the general criteria, is:

$$r(t_k, a_1, a_2, \ldots, a_N) = \sum_{i=1}^{N} w_i t_k \tag{2.38}$$

where the absolute value of w_i serves as an appropriately chosen penalty, which the i-th Agent is inflicted with, any time it exhibits either under-performing or over-performing behavior. A proper choice of w_i may be the following :

- $w_i(t_k) = -100$ if $e_i(t_k) < -0.15$ (i.e., if severe under-performance is experienced by Agent i);
- $w_i(t_k) = -10$ if $0.15 < e_i(t_k) < 0$ (i.e., if minor under-performance is experienced by Agent i);
- $w_i(t_k) = -1$ if $0 < e_i(t_k) < 0.1$ (i.e., if acceptable over-performance is experienced by Agent i);
- $w_i(t_k) = -50$ if $e_i(t_k) > 0.1$ (i.e., if undesirable over-performance is experienced by Agent i).

In particular, the thresholds on the QoE Error values of w_i as below described have been arbitrarily chosen in order to suitably classify the behavior of Agent i at time t_k as a result of the joint action taken. Moreover, the initial policy, that is, the initial CoS-to-application association, is randomly generated.

The results will be showing as obtained in the described simulation scenario; in particular, the H-MARL-Q algorithm is applied with a number of Agents $N = 100$ and $N = 1000$, both in the High and Medium Traffic conditions. It should be emphasized that we can deal with such a high number of Agents due to the fact that the proposed H-MARLQ algorithm relies on a Reduced Joint Action Space, which has cardinality $S \times N = 300$ in the scenario with 100 Agents ($S = 3$ and $N = 100$), and $S \times N = 3000$ in the scenario with 1000 Agents ($S = 3$ and $N = 1000$). If the original Joint Action Space were used, a solution relying on the MARL-Q algorithm would be unfeasible, since the cardinality would be $S^N = 3^{100} = 5.2 \times 10^{47}$, and $S^N = 3^{1000} = 1.42 \times 10^{477}$ in the two scenarios, respectively. The results obtained with the H-MARL-Q algorithm are compared with the performance of a Static algorithm which adopts a static CoS assignment policy. The comparison with the MARL-Q algorithm is impossible due to the curse of dimensionality. The obtained results are expressed in terms of two quantities:

- The Average Absolute QoE Error, computed as the absolute value of the QoE Error, averaged over all the considered Agents and all the simulation episodes;
- The QoE Error Standard Deviation, computed as the standard deviation of the QoE Error vector $(e_1, e_2, \ldots, e_N)$ (where e_i, for $i = 1, 2, \ldots, N$) averaged over all the simulation episodes

Note that the standard deviation accounts for the fairness among Agents: the smaller the standard deviation, the higher the fairness among Agents. Figs. 2.4-2.7 clearly show that the H-MARL-Q algorithm remarkably outperforms the Static algorithm in all of the considered simulation cases. In particular, while under the Static algorithm the Average Absolute QoE Error is appreciably smaller in Medium rather than in High Traffic conditions, under the H-MARL-Q algorithm, for both $N = 100$ and $N = 1000$, the Average QoE Error bars corresponding to High and Medium Traffic conditions (see Figs. 3 and 4) exhibit values that are really close to each other: this means that the presented algorithm also allows to overcome the disadvantages related to the impact that the traffic congestion conditions produce on the bottleneck link. Furthermore, the QoE Error Standard Deviation shown in Figs. 2.6 and 2.7 confirms the virtues of the H-MARL-Q algorithm, since the dispersion of the QoE Error values of the different Agents at the end of the learning procedure is significantly closer to zero than in the case when the Static algorithm is applied.
All these results evidently show that the dynamic and personalized selection of the most appropriate CoS for the ongoing application instances yields improved performance results, if compared with a static CoS assignment policy. In addition, Fig. 2.8 shows the Average Absolute QoE Error trend, i.e., the evolution of the Average Absolute QoE Error over time.
Let the settling time denote the time needed by the Average Absolute QoE Error to reach a steady state. Once an acceptable preliminary agreement among Agents – yielding the selection of the most "promising" joint actions for solving the dynamic CoS assignment problem – has been reached in step (a), the error dynamics, as highlighted in Fig. 2.8, experiences a rapid decrease over the first 100 iterations of step (b) and then it takes some time to settle down to the steady-state value: in the figure, the settling time is approximately equal to 9000 iterations. So, the overall runtime required by the H-MARL-Q algorithm is the sum of the time t_a necessary to reach the preliminary agreement in step (a) plus the time t_b necessary to perform step (b), where t_b amounts to

approximately 9000 iterations for $N = 100$ and t_a is negligible with respect to t_b. This is indeed an encouraging result which shows that the H-MARL-Q algorithm has to be preferred to the MARL-Q algorithm as the former achieves a satisfactory approximate solution in a reasonably smaller amount of runtime than the latter – whose runtime, instead, actually turns out to be unfeasibly long in scenarios where the number of Agents is counted in the order of hundreds or thousands.

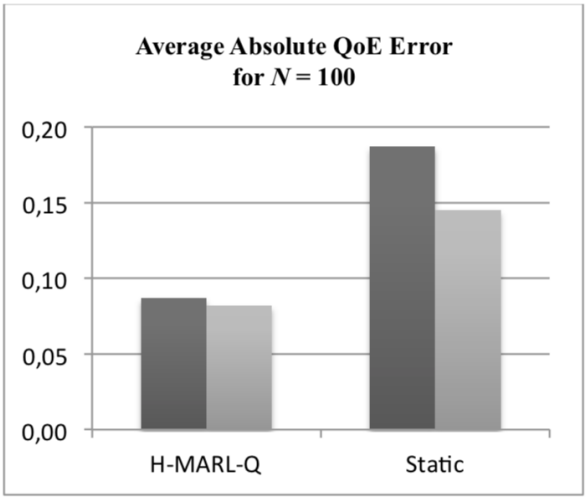

Fig. 2.4 Average Absolute QoE Error for $N = 100$. The dark-grey bar and the light-grey bar represent the Average Absolute QoE Error in High and Medium Traffic conditions, respectively.

The proposed approach to QoE Control enables a dynamic Class of Service [81] selection aimed at reducing the error between the personalized Perceived QoE and the personalized Target QoE levels by properly driving the control procedures that handle the underlying networks. This result could be obtained by embedding a new Multi-Agent Reinforcement Learning algorithm, namely the proposed H-MARL-Q algorithm, in a centralized QoE Controller.
The proposed method presents several practical advantages:

- it does not require any a-priori knowledge of the environment (i.e., it is model-free) thanks to the adoption of a Reinforcement Learning based approach;
- it is decoupled from QoE Evaluation, i.e., it can work in conjunction with any algorithm computing the Target QoE and the Perceived QoE

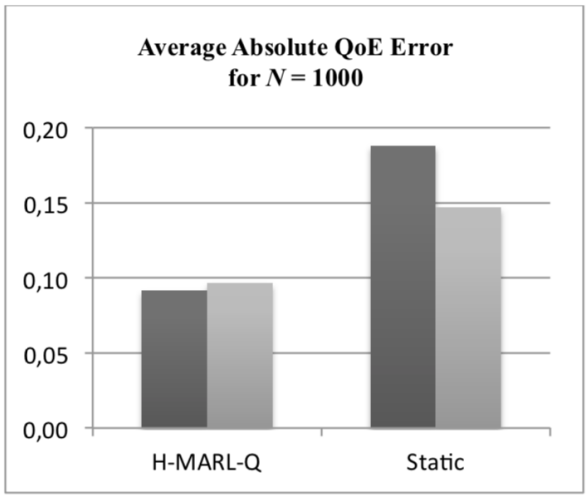

Fig. 2.5 Average Absolute QoE Error for $N = 1000$. The dark-grey bar and the light-grey bar represent the Average Absolute QoE Error in High and Medium Traffic conditions, respectively.

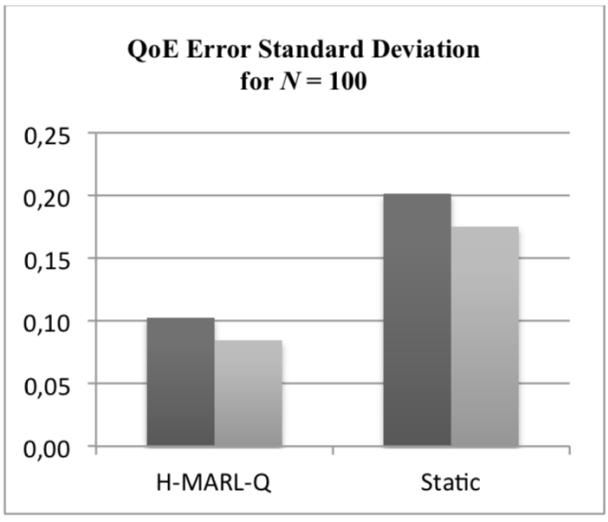

Fig. 2.6 QoE Error Standard Deviation for $N = 100$. The dark-grey bar and the light-grey bar represent the QoE Error Standard Deviation in High and Medium Traffic conditions, respectively.

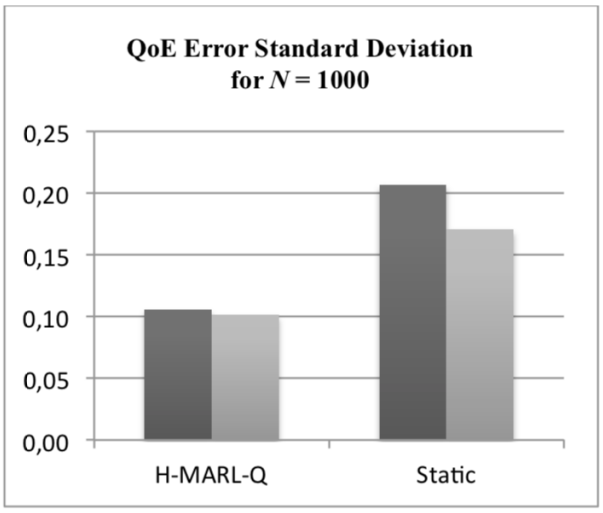

Fig. 2.7 QoE Error Standard Deviation for $N = 1000$. The dark-grey bar and the light-grey bar represent the QoE Error Standard Deviation in High and Medium Traffic conditions, respectively.

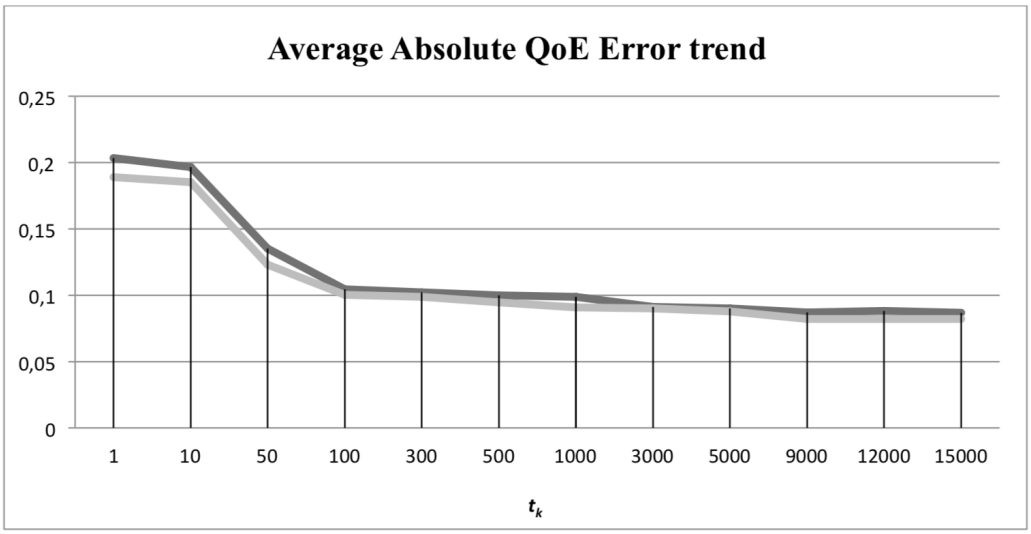

Fig. 2.8 Average Absolute QoE Error trend, corresponding to step (b) of the H-MARL-Q algorithm, in High (black line) and Medium (grey line) Traffic conditions with N = 100.

values, and it allows a personalization level up to the single application instance, since the only signal exchanged at the interface between the QoE Controller and the QoE Evaluator is the QoE Error provided by 2.29;

- it requires minimal signaling overhead since no communication exchange among Agents is needed and very little information has to be exchanged among the centralized QoE Controller and the distributed Agents;
- it is characterized by a very good degree of scalability (thus being able to handle several hundreds of Agents) due to the fact that, as the joint action to be carried out at each time step is sought within a suitable Reduced Joint Action Space, the complexity of the proposed H-MARL- Q algorithm is linear in the number of Agents (as opposed to the well-known MARL-Q algorithm whose complexity is exponential in the number of Agents).

At present, the authors are carrying out further research, based on consensus in networked dynamical systems, with the aim of overcoming the centralized paradigm and, consequently, of developing a solution in which the QoE Control functionalities are fully distributed into the Agents.

2.4 RL Control approach applied to transportation system

Some content of this section is part of the publication in [17].

Smart data aggregation, transmission and analysis are key features in Intelligent Transportation System (ITS). Since the beginning, these mobility support systems allowed both urban, local trip planning and control solutions integrating private and public transportation means, and door-to-door long distance multimodal journey planning with the aim of optimizing specific travel aspects (e.g., cultural visit, low emission, etc.). Due to the wide deployment and large range of applications, ITSs can take advantage of the most advanced ICT architectures and technologies. Among them, Future Internet (FI) offers one of the most promising frameworks for efficient, large-scale solutions. In this work, we present a FI oriented system for a closed-loop, control-based approach driving existing ITSs in personalizing basic services for end users. The proposed approach is user centric, in the sense that it is sufficiently general to allow the personalization of a family of services, ranging from trip planning and control services to tariff design. In particular, we consider the personalization of a multi-modal trip planning service.

Nowadays ITSs represent an important topic from the technological and business point of view. They mainly rely on models and algorithms from Transportation Engineering that are deeply changing our travel habits. The main goal is that of making current transportation systems more and more safe, secure and efficient, as well as providing intelligent multi-modal trip plans and reducing risks, traffic congestion, and CO2 emissions. From the ICT point of view, ITSs take advantage of real-time information and communication for supporting end user decision-making [93]. In this respect, an ITS can be considered as intelligent since it combines several disciplines including, but not limited to, effective and efficient optimization and control algorithms, as well as computer science innovative services and advanced architectures. According to [114], from the functional point of view, an ITS can be divided into six fundamental components (see 2.9): (i) Advanced Traveler Information System (ATIS), (ii) Advanced Transportation Management System (ATMS), (iii) Advanced Vehicle Management (AVM), (iv) Business Vehicle Management (BVM), (v) Advanced Urban Transportation System (AUTS), and (vi) Advanced Public Transportation System (APTS). Among them, ATIS represents one of the most

important components in ITS, especially because it provides travelers with suitable real-time detailed information about journey, traffic status, public transport information, including time table and current availability of each transport means. From the conceptual point of view, ATIS consists of two main modules [111]: (a) Pre-Trip Traveler Information System (PTTIS) and (b) En- Route (also known as On-Trip) Traveler Information System (OTTIS). In applications of practical interest, ATIS can include the Multi-modal Trip Planner (MTP) module (see [78] - [34]). In order to provide suitable real-time information about travel, usually data coming from several, heterogeneous sensors are gathered and subsequently processed into the Advanced Transportation Management System (ATMS) [93]. A real challenge in ITS is integrating different, heterogeneous and dynamic data sources and providing information for each end user in a multi-modal transport model that could cover long distance journeys [113]. A highly challenging feature in ITSs is the personalization

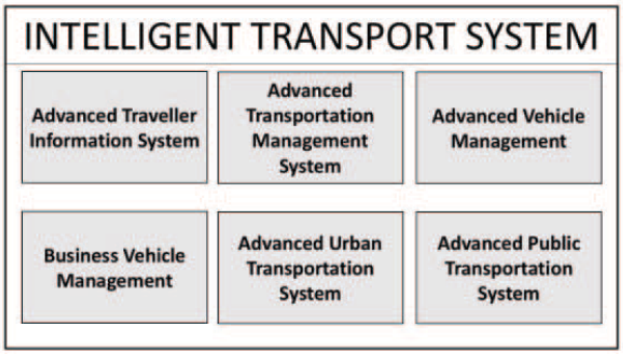

Fig. 2.9 Intelligent Transportation System

of real-time information, aimed at providing end users with suitable, on-line support taking into account user (i) requirements, (ii) preferences, and (iii) behavioral profile. Several approaches have been considered, ranging from Recommendation Systems to advanced Expert Systems. In [78], an advanced trip planner is presented aimed at providing personalized user information and travel alternative path. A personalized information retrieval system for filtering travel results, such as to satisfy user specific needs, is conceived in [35]. In [76], a multi-criteria decision making approach is proposed and validated in order to help, in a personalized way, the end user in the journey selection among several different travel solutions. In [10], a framework integrating Case Based Reasoning and Multi Criteria Decision Making is proposed in order to improve the user satisfactions for itinerary search in urban area. Three distinct

methods aimed at predicting journey times and ranking typologies of Points of Interest (i.e., stations, museums, etc.) according to user interests are proposed in [61]. Sophisticated machine learning methodologies have been proposed for

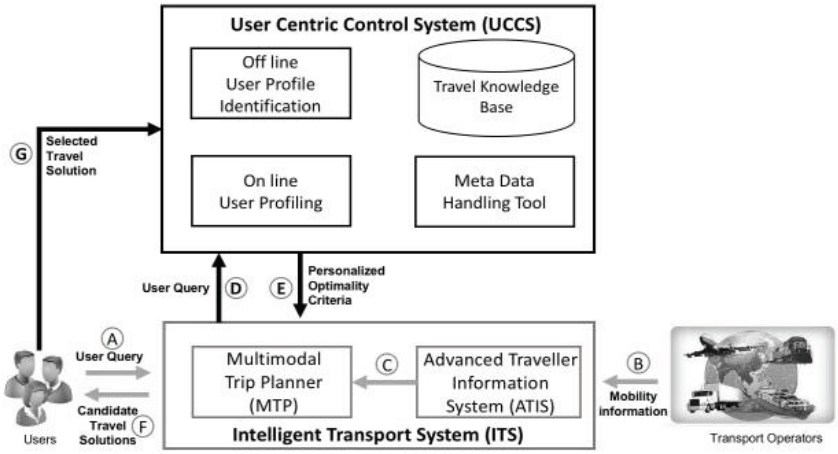

Fig. 2.10 Extended Intelligent Transportation System

user profiling. A Markov Decision Process and Reinforcement Learning based approach is proposed in [70] to learn how to support users in decision-making process through human-computer interaction. Bayesian methods have been adopted in [3] in order to learn user travel preferences by considering user past travel choices.

We propose an Extended Intelligent Transportation System (ExITS) whose key difference, with respect to previous control and/or learning based approaches for ITS, consists in jointly taking into account the user request submitted to the Multi-modal Trip Planning (MTP) module and the actual choice carried out by the user as one or more Candidate Travel Solutions are returned. As a matter of fact, these choices are elaborated by a dedicated User Centric Control System (UCCS) in order to refine the user behavioral profiling and eventually producing the so-called Personalized Optimality Criteria which drive the basic ITS to provide personalized travel solutions representing the Personalized Control Decisions as explained in Sections IV. The UCCS extends traditional ITS functionalities by automatically learning user preferences from the user behavior, even in the case the user preferences are in contrast with the ones explicitly declared.

2.4.1 Reference scenario

The transport network is modeled as a multigraph $G(V, E)$, namely a graph with possible multiple edges between the same vertices. A multigraph can be described by a function f: $E \to V \times V$ indicating the vertices (v_i, v_j) connected by a given oriented edge el, i.e. $f(e_l) = (v_i, v_j)$. In our model, there exists an oriented edge el connecting the vertices (v_i, v_j) if there exists a transport mean m directly linking the node v_i with the node v_j starting at time t_{il} from the node v_i and arriving at time t_{jl} at the node v_j. We assume that there are M distinct transport means. A transit node in V is a node v such that v is the source node of a link e served by a transport means m_i and the destination node of a link e served by a transport means m_j for some i,j and i, j $\{1, ..., M\}$. By referring to 2.10, the ITS includes the so-called Advanced Travel Information System (ATIS), whose main task is to keep dynamically updated the structure of the above-mentioned multigraph $G(V, E)$ on the basis of the real-time mobility information provided by Transport Operators. As a matter of fact, $G(V, E)$ is a basic input for the Multimodal Trip Planner (MTP) module. In this paper, we assume that U users are registered in the ITS platform and have subscribed the trip planning service. This service is used by user u whenever u needs to plan a door-to-door travel, i.e. a complete trip from a specific source location a to a target destination location b. The source and destination locations of the travels are two particular vertices of $G(V, E)$. We indicate by $d(a, b)$ the Euclidean distance between two dimensional vectors of GPS coordinates of physical locations a and b. User u can indicate the time t_a for travel to begin, t_a can be a specific time in a day, a generic daytime or a range of time in a day. Additionally, user u can provide a set of User Constraints, indicated by $UC(u, a, b, t_a)$, for the travel from a to b starting at time t_a. Typical constraints concern the number of passengers traveling with u, special needs indicated by u, allowed transport means among the possible M transport means, etc. We consider S possible special needs that user u can declare. Finally, user u can indicate an explicit set $UP(u, a, b, t_a)$ of User Preference Criteria about the desired travel. The criteria $UP(u, a, b, t_a)$ are provided in order to explicitly guide the MTP in proposing the most suitable travel solutions. In this work, we consider a basic family of criteria consisting of:

- minimizing the overall travel time;
- minimizing the overall travel cost;

- maximizing comfort level during the journey
- maximizing the class category of each transport modality;
- minimizing CO2 emissions during the trip;
- minimizing the number of transit nodes;
- minimizing the walking distance;
- minimizing the number of distinct modality means

These so called criteria represent a subset of a more general set of Optimality Criteria, here indicated as $OCR = \{oc_1, ..., oc_P\}$, consisting of all optimality criteria considered by MTP. In general, the cardinality P of OCR depends on the specific MTP and can be a very high number, often including particular combinations of criteria (e.g., convex combinations). We refer to as User Query, indicated as

$$Q_i(u) = (u, a, b, t_a, UC, UP) \tag{2.39}$$

the i-th query submitted by the user u who wants to plan the travel from the source location a to the destination location b starting at time t_a with the User Constraints $UC(u, a, b, t_a)$ and the User Preference Criteria $UP(u, a, b, t_a)$. For the sake of readability, we will refer to a generic i-th User Query $Q_i(u)$ as $Q(u)$. A Travel Sequence related to $Q(u)$, indicated as $s(Q(u))$, consists of a list of L distinct and consecutive edges e_l, (l_L) linking the source a with the destination b in such a way that $f(e_l) = (v_i, v_j)$, $f(e_{l+1}) = (v_j, v_l)$, $t_{jl} < t_{jl+1}$, $f(e_1) = (a, v_j)$ and $f(e_L) = (v_j, b)$. A Travel Sequence $s(Q(u))$ is referred to as Feasible if $s(Q(u))$ respects all constraints $UC(u, a, b, t_a)$. A Feasible Travel Sequence $s(Q(u))$ is Optimal if $s(Q(u))$ optimizes some Optimality Criterion in OCR. Note that, for processing time constraints, MTP cannot consider all the P possible Optimality Criteria; conversely, MTP will consider a suitably selected subset of OCR including, at the minimum, $UP(u, a, b, t_a)$. This subset will be hereinafter referred to as the set $SOCOCR$ of Selected Optimality Criteria. A Feasible Travel Sequence $s(Q(u))$ being relevant to a given $Q(u)$ and optimal according to some optimality criterion in SOC identifies a Candidate Travel Solution for $Q(u)$. We assume that MTP is able to find the entire set of Candidate Travel Solutions for $Q(u)$ on the basis of the set SOC and of the information related to the present structure of $G(V, E)$. Let $TS(Q(u)) = \{s_1(Q(u)), s_n(Q(u))\}$ be the set of Candidate Travel Solutions for

$Q(u)$ returned by MTP. Typically, the cardinality n of $TS(Q(u))$ increases as the number M of transport modalities gets larger and/or the distance $d_0(a,b)$ increases. We define Selected Travel Solution and indicate by $s^*(Q(u))$ the Candidate Travel Solution in $TS(Q(u))$ which is actually selected by the user u. We assume that the user u can be assigned the User Profile grouping all users being more similar to u with respect to three families of parameters: the distance $d_0(a,b)$, the constraints $UC(u,a,b,t_a)$ and the criteria $UP(u,a,b,t_a)$. The identification of K possible User Profiles is a task demanded to a specific functionality of the $UCCS$. Now, we introduce three important functions:
- $ocr : s \rightarrow OCS$ is the function indicating the optimality criterion $ocr(s)$ in OCR considered by MTP to provide the Candidate Travel Solution $s(Q(u))$ in $TS(Q(u))$;
- $prl : u \rightarrow prl_1, prl_2, prl_K$ is the function assigning a user u to her/his User Profile $prl(u)$ properly selected in a set of K possible User Profiles;
- $prs : prl_k$ is the function identifying the subset of OCR being suitable for the User Profile prl_k; in other words, for each possible prl_k for $k = 1, K$ the function prs allows the identification of the most relevant optimality criteria in the User Profile prl_k.

2.4.2 Extended Intelligent Transportation System

In this section, the ExtePOnded Intelligent Transportation System (ExITS) is outlined with respect to two working modes: the Basic Mode and the Cognitive Mode. In the following, we will refer to 2.10 in order to illustrate these working modes and to clarify the main differences between them. Both working modes are based on a basic ITS consisting of the main components ATIS and MTP. Both working modes allow each user u to submit a User Query $Q(u)$ in the pre-trip phase to the MTP. In both working modes, MTP is in charge of selecting the set $TS(Q(u))$ of n Candidate Travel Solutions which is returned to the user u 2.10). The MTP performs such selection on the basis of $Q(u)$ (arrow A) $G(V,E)$ (arrow C) Multigraph information is dynamically elaborated by the ATIS on the basis of the real-time Mobility Information received from Transportation Provider (arrow B). Note that the dynamic information included in $G(V,E)$ allows the MTP to react, in real-time, with respect to unforeseen events affecting the mobility dynamics. The Basic Mode, differently from the Cognitive Mode, does not makes use of the User Centric Control System (UCCS). Therefore, with reference to Figure 2, the information flow of

the Basic Mode just foresees the sequence A B C and F.
Conversely, the Cognitive Mode makes use of the UCCS including two fundamental functionalities being in charge for the actual personalization of mobility services: (i) the off-line User Profile Identification functionality and (ii) the on-line User Profiling functionality. In the Cognitive Mode, $Q(u)$ submitted by the user u (arrow A) is forwarded to the UCCS (arrow D).
As the UCCS receives $Q(u)$, the User Profiling functionality identifies, in real-time, the User Profile $prl(u)$ the user u belongs to in the set of K possible User Profiles. The identification of this set is demanded, off-line, to the User Profile Identification functionalities. Then, the User Profiling functionality identifies, via the $prs(prl_k)$ function when $prl_k = prl(u)$, the Personalized Optimality Criteria $POC(Q(u)) \subseteq OCR$, associated to the User Profile $prl(u)$, i.e., optimality criteria being more likely preferred by the user u when traveling from a to b starting at time t_a as indicated in $Q(u)$. In general, the cardinality of $POC(Q(u))$ will be much lower than the cardinality of OCR.
The set $POC(Q(u))$ of Personalized Optimality Criteria are provided by the UCCS to MTP (arrow E). Accordingly, MTP can calculate the set $TS(Q(u))$ of Candidate Travel Solutions with respect not only to the explicit User Preference Criteria $UP(u, a, b, t_a)$ (as in the Basic Scenario, arrow A), but also to the implicit Personalized Optimality Criteria $POC(Q(u))$ (arrow E). In other words, in the ExITS, the set SOC includes both the set of User Preference Criteria and the set of Personalized Optimality Criteria, i.e. $SOC = UP(u, a, b, ta) \cup POC(Q(u))$.
Therefore, the MTP considers both preferences explicitly indicated by the user u in $Q(u)$ and the Personalized Optimality Criteria selected by the $UCCS$ as user profile based preferences implicitly inducted by the machine learning approach.
In the Cognitive Mode, the set $TS(Q(u))$ of Candidate Travel Solutions, computed by the MTP according to SOC, is returned to user u (arrow F) once ranked according to the following rule. The first Candidate Travel Solutions appearing in the ranked list are the ones satisfying the explicit User Preference Criteria $UP(u, a, b, t_a)$. Instead, the following Candidate Travel Solutions satisfy the implicit Personalized Optimality Criteria $POC(Q(u))$. Once the ranked set $TS(Q(u))$ of Candidate Travel Solutions is returned to user u, such a user makes her/his choice by selecting one of Candidate Travel Solutions in $TS(Q(u))$, i.e., the Selected Travel Solution $s^*(Q(u))$. It is fundamental

to note that, differently from the Basic Mode, the Selected Travel Solution $s^*(Q(u))$ is provided to the UCCS (arrow G), in order to be used as a feedback of paramount importance in the machine learning algorithms embedded in the User Profiling Identification module. Therefore, with reference to figure 2.10, the information flow of the Cognitive Mode foresees the complete sequence from A to G.
It is important to remark that the Cognitive Mode is fully in line with the Future Internet concept of considering a service/technology independent Controller (here represented by the MTP). In fact, the Controller takes control decisions (here represented by the set $TS(Q(u))$ of Candidate Travel Solutions) on the basis of the feedbacks directly coming from monitoring of the present system status (here represented by the User Query $Q(u)$ and the Mobility Information), according to a control law depending on some user centric, personalized Driving Parameters (here represented by the Personalized Optimality Criteria $POC(Q(u))$).

Within the UCCS, four main functional modules are in charge of dealing with User Query $Q(u)$ at the aim of generating the Personalized Optimality Criteria $POC(Q(u))$.
The Travel Knowledge Base has the role of storing all additional data considered in the Cognitive Mode but not in the Basic Mode. These interactions are represented by a set HP of records $R_i(u)$ of the form

$$R_i(u) := [Q_i(u), POC(Q_i(u)), ocr(s^*(Q_i(u)))]^T \tag{2.40}$$

Each $R_i(u)$ includes the i-th User Query $Q_i(u)$ submitted by user u, the Personalized Optimality Criteria $POC(Q_i(u))$ returned by UCCS with respect to $Q_i(u)$, and the optimality criterion $ocr(s^*(Q_i(u)))$ considered by the MTP when $s(Q_i(u))$ is the Selected Travel Solution $s^*(Q_i(u))$. The Travel Knowledge Base stores records having the structure (see equation 2.40). A new record of this structure is created whenever a Candidate Travel Solution in $TS(Q_i(u))$ is selected by u. The set HP of historical records are provided to the User Profile Identification module so that the set of K User Profiles can be calculated off-line. In addition, when a new $Q(u)$ triggers the UCCS (arrow D), the User Profiling module detects on-line the User Profile $prl(u)$ associated to user u on the basis of the information stored in the Travel Knowledge Base. Accordingly, Personalized Optimality Criteria $POC(Q(u))$ are provided to the MTP via the prs function (arrow E). Once the Selected Travel Solution $s^*(Q(u))$ is chosen

in $TS(Q(u))$ by the user u (arrow F), a new record $R_i(u)$ is inserted in the Travel Knowledge Base.
The Meta Data Handling Tool offers basic procedures to gather, store and manage data records of the form (2.40) in the Travel Knowledge Base. This module is in charge for extracting the features of a given $Q(u)$, namely the distance $d_0(a, b)$, the vector $UC(u, a, b, t_a)$ and the vector $UP(u, a, b, t_a)$, as made available by the ITS (arrow D). Moreover, this module is in charge of transmitting the set $POC(Q(u))$, identified by the User Profiling, to MTP. Finally, the Meta Data Handling Tool implements the function $ocr : s \to OCR$ indicating the optimality criterion $ocr(s)$ considered by the MTP for Candidate Travel Solution s.
The proposed ExITS is sufficiently general in the sense that UCCS can adapt to any MTP by means of the Metadata Handling Tool. In fact, every MTP accepts User Query $Q(u)$ containing at least the source a and the destination b. In this elementary case, $UC(u, a, b, t_a)$ and $UP(u, a, b, t_a)$ are empty vectors and the only data that the Meta Data Handling Tool can extract from $Q(u)$ is the distance $d_0(a, b)$. Accordingly, other components in UCCS can work seamlessly.
The User Profile Identification is the core module of UCCS. In fact, this module is able to analyze a huge amount of historical records HP of the form (2.40) stored in the Travel Knowledge Base and to identify the set of K User Profiles that will constitute the basic family of user profiles driving the User Profiling module. The automatic identification of the User Profiles is made on the basis of the similarity between historical records $R_i(u)$ in HP. The main idea is that similar User Queries $Q(u)$ and $Q(w)$ should be characterized by similar sets $TS(Q(u))$ and $TS(Q(w))$ of Candidate Travel Solutions, that means similar Personalized Optimality Criteria $POC(Q(u))$ and $POC(Q(w))$. Note that the Personalized Optimality Criteria $POC(Q(u))$ are proposed by the ExITS on the basis of historical patterns of User Query being similar to $Q(u)$. On the contrary, the Selected Travel Solution $s^*(Q(u))$ is explicitly chosen by the user u and, therefore, the optimality criterion $ocr(s^*(Q(u)))$ represents the actual, implicit preference of user u with respect to $Q(u)$. The User Profile Identification algorithm for determining the Personalized Optimality Criteria $PCP(Q(u))$.
The User Profiling module implements two distinct functions. The former is the function prl: $u \to \{prl_1, prl_2, \ldots, prl_K\}$ assigning a user u to User Profile $prl(u)$

in the set of K possible User Profiles already identified by the User Profile Identification module. The latter is the function prs:$\{prl_1, prl_2, \ldots, prl_K\} \rightarrow \{0,1\}^P$ identifying the subset of OCR being suitable for each User Profile prl_k for $k = \{1, .., K\}$.

2.4.3 Data Driven User Profiling

In this section, functionalities of the User Centric Control System (UCCS) (see 2.10) are described from the methodological and algorithmic point of view. These functionalities rely on similarity based behavioral profiling techniques typically used to learn user preferences by analyzing a huge amount of data records and inferring similar behaviors. Data describing both user preferences and behaviors with respect to the considered service, in this case the MTP, are a key aspect of the analysis.

The complete set of entries of record $R_i(u)$ is summarized in Table 2.2. It is important to remark that all entries are numerical values (real, integer or Boolean). Of course, according to the specific features of the considered MTP, some sub-vectors in the User Query $Q(u)$ can be available or not. In this case, the Meta Data Handling Tool will not manage these data and the record stored in the Travel Knowledge Base will not include the resulting missing data with no consequence for the machine learning algorithms.

Analyzing the feedback provided by users with respect to a given service is fundamental for user profiling and service personalization. In this work, we consider the mobility service offered by MTP integrated in the ITS via ATIS, as explained in the reference scenario. In this case, the explicit feedback provided by the user u is the Selected Travel Solution $s^*(Q(u))$ actually chosen by the user u in the set $TS(Q(u))$ of Candidate Travel Solutions returned by the MTP. We are interested in defining a finite number K of User Profiles describing general, but recurrent patterns of user behaviors when users choose $s^*(Q(u))$ in $TS(Q(u))$. In order to identify the set of possible User Profiles, we consider a pattern of the form

$$I(Q(u)) := [d_0(a,b), UC(u,a,b,t_a), UP(u,a,b,t_a), ocr(s^*(Q(u)))]^T \quad (2.41)$$

We consider that for each user request, i.e., for each interaction between an user u and the ExITS producing a User Query $Q(u)$, a record $R_i(u)$ (2.40) is

Table 2.2 Data Record

Vector	Entry	Description	Type
$Q(u)$	u	User identifier	Integer
	a	Source Location	GPS
	b	Destination Location	GPS
	t_a	Begin travel time	Integer
	$UC(u,a,b,t_a)$	Owned transport mean (Yes or No)	Boolean
		Fidelity Program	Boolean
		Reason for traveling	Boolean
		Number of Passengers	Integer
		Maximum number of transit nodes	Integer
		Requested return (Yes or NO)	Boolean
		Vector of allowed types of transportation mean	{0,1}^M
		Preferred class category	Integer
		Maximum allowed overall price	Real
		Allowed total travel time	Integer
		Vector of special travel request	$\{0,1\}^S$
	$UP(u,a,b,t_a)$	Minimize the overall travel time	Boolean
		Minimize the overall travel cost	Boolean
		Minimize comfort level	Boolean
		Maximize the class category	Boolean
		Minimize CO2 emissions	Boolean
		Minimize the number of transit nodes	Boolean
		Minimize the walking distance	Boolean
		Minimize the number of distinct modalities	Boolean
POC(Q(u))		Personalized Optimality Criteria for Q(u)	$\{0,1\}^P$
$ocr(s^*(Q(u)))$		Optimality Criterion explicitly selected by user u	$\{0,1\}^P$

stored in the Travel Knowledge Base and a pattern I(Q(u)) can be used for user feedback analysis by the User Profile Identification. The user feedback analysis consists of a partitional clustering procedure that, given a set HP of historical patterns $I(Q(u))$, yields a partition $\prod(HP) = \{W_1, ..., W_K\}$ of K non empty subsets $W_i \subset HP$ such that $\bigcup_{t=1,...,K} W_t = HP$ and $W_t \cap W_j = \emptyset$ for each $i, j = 1, ..., K$ and $i \neq j$. The K components of partition $\prod(HP)$, also called clusters, represent groups of similar patterns with respect to an inter cluster separation criterion or, alternatively, an intra cluster homogeneity criterion ([40]). Both approaches rely on a priori selected distances (typically lx-norm metrics, e.g. [108]) measuring the inter cluster similarity or the intra cluster dissimilarity, respectively. According to the criterion to optimize, a number of partitional clustering algorithms have been proposed in the literature. Some algorithms are indicated in case of numerical attributes describing the patterns to be clustered, others are more suitable when dealing with mixed attributes. The most of clustering procedures consider the number K of clusters as input parameter to the procedure (e.g., k-means, see [69] and [43]). There exist approaches not requiring the number of clusters as input parameter (e.g., Clique Partitioning Problem, see [7]). The selection of the suitable partitional clustering algorithm depends on the characteristics of the user profiling problem. In this work, we consider patterns $I(Q(u))$including all numeric attributes (see Table 2.2), and adopt the k-means algorithms for selecting the optimal partition in k clusters by minimizing the so-called Sum of Squares Error (SSE), i.e. the sum of the squared Euclidean distances between each pattern $I(Q(u))$ and the cluster centroid. This choice is particularly suitable, since the Euclidean distance allows capturing the differences between different patterns, once their entries are normalized (in our case, we consider a normalization in $[0, 1]$). The selection of parameter k of the clustering algorithm is a key feature of our approach. In fact, even if the clustering procedure is completely unsupervised, we deal with the problem of selecting the suitable number K of clusters by analyzing the results of the clustering obtained with different values of the input parameter k and by selecting the value K corresponding to the minimum number of distinct optimality criteria $ocr(s*(Q(u)))$ represented in each cluster. In such a way, similar users belong to same cluster W_i and share the same user profile given by the centroid of W_i. The user profile is characterized by the most representative optimality criteria appearing in the cluster W_i that are in a restricted number of meaningful user profile based optimality criteria. The

representative optimality criteria in cluster W_i are the ones returned as the set $POC(Q(u))$ of Personalized Optimality Criteria by the on-line User Profiling module for all $Q(u)$ such that $I(Q(u))$ is similar to the centroid of the cluster W_i. In this sense, we say that Personalized Optimality Criteria returned by the User Profiling module are personalized.
The output of the off-line User Profile Identification module is the set $\prod(HP) = \{W_1, ..., W_K\}$ of clusters of homogeneous patterns $I(Q(u))$. For each cluster W_k, the centroid of the cluster is given by the mean values of entries of patterns $I(Q(u))$ belonging to W_k. In the following, for each $k = 1, .., K$, we indicate by prl_k the sub vector

$$prl_k = [d_0(a, b, UC(u, a, b, t_a), UP(u, a, b, t_a)]^T \tag{2.42}$$

where UC (i.e. vector) and a in UP, including the mean values of entries $d_0(a, b)$, $UC(u, a, b, t_a)$ and $UP(u, a, b, t_a)$ of all patterns $I(Q(u))$ belonging to W_k. It is important to remark that a single user can belong to different clusters. In fact, if two User Queries are assigned to distinct clusters, then the user who submitted the queries behaved differently and, accordingly, his behaviors have been assigned to distinct groups in $\prod HP$.
Given the output of the off-line User Profile Identification, namely the set $\prod HP = \{W_1, ..., W_K\}$ and the related sub vectors $\{prl_1, ..., prl_K\}$ (2.42), we denote by $J(Q(u))$ the sub vector of the pattern $I(Q(u))$ given by

$$J(Q(u)) = [d_0(a, b), UC(u, a, b, t_a), UP(u, a, b, t_a)]^T \tag{2.43}$$

Once a new User Query $Q(u)$ is submitted by a user u to the MTP, the on-line User Profiling procedure evaluates the cluster $prl(u)$ which $Q(u)$ belongs to. In order to do that, the Euclidean distance between the sub vector $J(Q(u))$ and each sub vector prl_k of the K centroids of the partition $\prod HP$ is calculated. $Q(u)$ is assigned to the cluster whose centroid is the closest to $J(Q(u))$.
In this sense, the User Profiling module implements the function $prl : u \rightarrow \{prl_1, prl_2, \ldots, prl_K\}$ by assigning to the user u the cluster prl_k^* obtained as follows

$$K^* = argmin_{k=1,...K} d_{l_2}(J(Q(u)), prl_k) \tag{2.44}$$

In order to provide the Personalized Optimality Criteria $POC(Q(u)) \subset OCR$, the User Profiling module implements the function $prs : \{prl_1, prl_2, \ldots, prl_K\} \rightarrow \{0, 1\}^P$ that identifies the subset $POC(Q(u))$ of the P possible Optimality

Criteria being suitable for a given User Profile prl_k for $k = 1, .., K$. In order to do that, given the User Profile $prl(u)$ assigned to User Query $Q(u)$ by (2.43), the function $prs(prl(u))$ returns the subset of OCR being mostly represented in cluster $prl(u)$ in terms of high percentage of patterns $I(Q(u))$ sharing the same selected optimality criterion $ocr(s^*(Q(u)))$.

2.4.4 User Centric Control System

I have realized a questionnaire to collect user data in terms of users general preferences and needs during journeys. The answers from survey participants have been used to implement a data driven user profile approach by training a K-means algorithm (a Machine Learning unsupervised algorithm) able to divide in k clusters/groups the travelers as described in the previous sections. The result of having divided in K clusters/groups the users has carried out different user profiles (as presented above) according to questionnaire answers. By analyzing the main users preferences and needs through the computed user profiles I have defined a **traveler (user) model**, which represents a synthetic description of a generic user in the travel context with a set of different travel characteristics.
The identification of a traveler model is the starting point to enable a learning process via *human-machine interaction*.
In the human-machine interaction, i.e. when a human being interacts with an application which provides services, a learning system can be setup (see figure 2.11) able to intercept the human behavior/choices during the interaction with the application.
To intercept and learn from human behavior the learning system has been designed to include the traveler model, so that it is able to learn the human choice or actions, according to the traveler model, while the human begin interacts with the application.
The traveler/user model is characterized by 11 different characteristics (TC):

- Number of transit nodes: representing the mean of number of modalities changes during performed journeys.
- Preferred transport means: representing the preferred modalities during each journey.

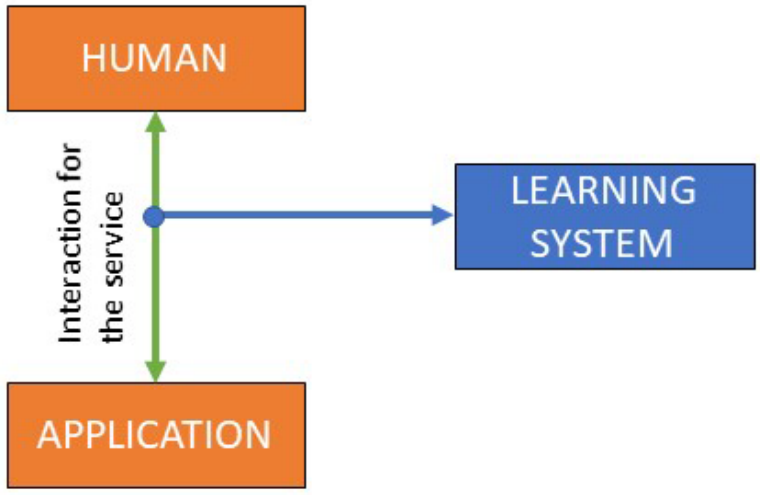

Fig. 2.11 Human Machine Interaction with learning system

- Private service car: representing the car-modalities (e.g., car rental, car sharing, car pooling, etc.) mostly used by the user for its journeys.
- Preferred class category: representing the class category most preferred by the user among the available class categories chosen performing its journey.
- Special travel needs: The travel needs are correlated to the class categories, since each category offers different comfort; in order to acquire preferred needs, this characteristic defines the exact needs preferred in choosing journey.
- Total travel time: each journey has a total time that each user has to respect to finish its trip; this characteristic attempts to define the total time admitted for each user.
- Comfort level: for each journey transport providers or private service discovery may define a comfort level for the corresponding journey.
- Sensibility to CO_2 emission: The sensibility to CO_2 emission can determine in a-priori manner the user choose. In particular, any model could drive a journey planner only consider the sensibility to CO_2 emissions, by computing only routes while considering this characteristic.
- Walking distance: The total distance admitted while user travels

- Range departure time: Each user could have a preferred departure time.
- Price range travel: representing the willingness to pay for each user during its journey.

The user model characteristics TC_i were used to acquire knowledge from users' expectation in order to determine perfect users' needs. In this respect, the Extended ITS presented in figure 2.10 were enriched by including an adaptive system as shown in figure 2.12.

The architecture presented in figure 2.12 extends basic functionalities offered

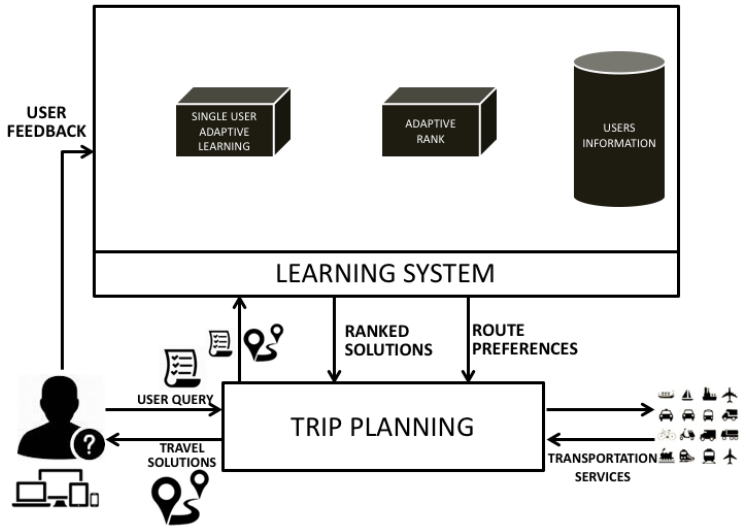

Fig. 2.12 Control Scheme for Intelligent Transportation System

by typical ITS modules or subsystem (already presented in figure2.10). The modules included in figure 2.12 are:

- *Trip Planning (TP)* typically included in the ATIS, is the module in charge of processing the user query in the pre-trip planning phase; it computes the trip solutions expected by each user as a consequence of a query.
- *The Single User Adaptive Learning* module in charge of iteratively learning user travel preferences and behavioral profile, exploiting the user cognitive model, where the model is designed as a set of travel features. It also provides to the *TP* module the route preferences for computing tailored solution.

- *The rank module* is in charge of providing the list of travel solutions ordered following the criteria provided by Single User Adaptive Learning.

By learning from human-machine interaction, I have used the traveler model with the purpose of creating an individual control experience, in charge of providing individual parameters for both (i) computing ordered personalized travel solutions and (ii) delineating knowledge about specific users' preferences and needs (also defined as behavioral context), for computing tailored solution. The learning system captures the travels' characteristics most preferred by users via explicit feedback (namely the *user feedback* as shown in figure 2.12). The feedbacks is then modeled over the traveler model permitting to outline the users behavioral context (how the users behave in choosing the travel solutions). More precisely, I have designed an algorithm that on the basis of the traveler model is able to learn which are (i) the preferred travel preferences/needs and (ii) the behavioral context contained in *Single User adaptive learning* (see 2.12) modules.
The travel preferences/needs refer to those travel characteristics selected by a generic user while choosing a travel solution considered as mandatory for performing a travel; conversely the traveler behavioral context refers to the travel characteristics that are not the most preferred by the user, but that are anyway chosen in the selected travel solution. The concept of behavioral context deserves a clear example: Imagine that, by using a generic journey planner system, a generic user makes its request for a long range travel where the departure location is *Rome* and the destination location is *Sydney*, as a consequence of the request the users receives a set of different solutions which include *airplane*; now it is possible to assert that for the requested travel the transport mean *airplane* is essential; but the generic user doesn't prefer the *airplane* transport mean for generic reasons; the travel is in some sense impossible without the airplane and the generic user picks the solution containing the airplane mean; this is the behavioral context, how the user behaves facing with those travels which contain some not preferred characteristics.
A generic system which doesn't include a learning system doesn't consider this selection as *special* selection, while a supervised system, described in the work, captures this selection and marks it as behavior. Obviously, figure out the difference between a choice that is either a consequence of the user preference or user behavior is not simple and it requires enough interactions (human-machine interactions) to be captured. When the user interacts with the system by

making decision in terms of selected travel solutions, the adaptive learning system captures the user choices and compares the picked solution with the other in the list; if the user picks solutions that don't contain *airplane* transport mean even if in some solutions it is contemplated the system could learn that this is a preference i.e. the user prefers *TRAIN* for instance; while selecting the solution that contain *airplane* transport mean among the other that don't contain any alternatives (the set of solutions includes only *airplane* mean) the adaptive learning system could capture the behavior taking into account the whole user's history i.e. it has always selected *train* mean and now it has selected *airplane* mean.

To implement an individual control experience the generic user has to be learned in its whole completeness, that is what he/she prefers and what he/she does not prefer but is in any case chosen.

For the sake of simplicity, within each 11 defined characteristic TC_i can be identified a set of different actions. The user preferences/needs and behavioral context are characterized by the more suitable actions in each characteristic TC for a given user, where the goal is to provide to each user the personalized ordered travel solutions evaluating what the algorithm has learned in terms of preferences and behavioral context.

The implementation of a such adaptive learning system is allowed by the Reinforcement Learning approach and the I have identified a stateless Q-Learning algorithm able to actualize such a adaptive learning system. The preferences/needs and the behavioral context that the adaptive learning system expects to learn concern those characteristics that are directly involved when users interact with an application able to satisfy the users' request with appropriate travel solutions.

The traveler model is characterized by a number of M characteristics, f_i, $i = 1, \ldots, M$ ($f_i = TC_i$). For each characteristic f, a set of distinct available actions a_n, $i \in A_n$ where $n = 1, \ldots, N_i$.

The overall action space $\mathbf{A}$ is computed as $\mathbf{A} = A_1 \times A_2 \times ... \times A_k$, with cardinality $N = \prod_{N=1,...,M} N_i$ i.e., it considers all the possible combinations of actions available in each characteristic. In this way, a set of vectors $\mathbf{a}_j \in \mathbf{A}$ is obtained, where $j = 1, \ldots, J$ represents the joint actions that must be evaluated to suggest the personalized ordered travel solutions describing also the behavioral context. Each joint action $\mathbf{a}_j$ contains the available action a_n for each n, implying that we are aiming at evaluating the optimal joint action,

defined as:

$$\mathbf{a}^* = argmax_{a \in A} Q(\mathbf{a}) \tag{2.45}$$

i.e., the action that fits better with the behavioral context of a given user. The action-value function update is:

$$Q_{t+1}(\mathbf{a}_t) = (1 - \alpha)Q(\mathbf{a}_t) + \alpha_t[r_t + \gamma max_{\mathbf{a} \in \mathbf{A}} Q_t(\mathbf{a})] \tag{2.46}$$

where $\mathbf{a}_t$ represents the joint action performed at time t, is the discount factor, r_t is the immediate reward and t is the learning parameter. The choice of α determines the convergence of the algorithm. An appropriate choice [75] is the following:

$$\alpha_t = 1/1 + visit_n(\mathbf{a}) \tag{2.47}$$

where $visit_n(\mathbf{a})$ is the number n of times that a given joint action $\mathbf{a}$ is used.
In practice, the number of characteristics of interest and the number of actions $\mathbf{a}_n$ in each joint action $\mathbf{a}_j$ are large enough to make the number J of joint actions a big obstacle for the learning process – this problem is known as the "curse of dimensionality" [14].
To avoid the hard limits of the learning process due to the already mentioned curse of dimensionality, the algorithm is designed by assuming that the characteristic are independent; then, a single-agent (SA) stateless Q-Learning algorithm is implemented for each characteristic f_i. The traveler model is represented as a collection of characteristic f_i, where $i = 1, \ldots, M$, for each characteristic f_i can be considered the set of actions $a_{n,i}$, $i \in A_i$. The characteristics (already discussed above as traveler model) and actions predicates are:

- f_1: number of transit nodes ($a_{n,1}$)
  $a_{n,1}$: The number of transit nodes (i.e., number of exchange nodes) that characterize the travel solutions.

- f_2: user transport means ($a_{n,2}$)
  $a_{n,2}$: The transport means that may be used during travels.

- f_3: private service car ($a_{n,3}$)
  $a_{n,3}$: The private transport means.

- f_4: preferred class category ($a_{n,4}$)
  $a_{n,4}$: The implied class categories used choosing several transports means.

- f_5: total travel time ($a_{n,5}$)
  $a_{n,5}$: The total travel time is evaluated dividing the interval between 0 and 1 in n subinterval.

- f_6: special requests ($a_{n,6}$)
  $a_{n,6}$: The additional services that may be requested during travels.

- f_7: walking distance during travel ($a_{n,7}$)
  $a_{n,7}$: The overall availability to walk during travels evaluates dividing the interval between 0 and 1 in n subinterval.

- f_8: sensibility to CO_2 emissions ($a_{n,8}$)
  $a_{n,8}$: The predisposition in choosing solutions pay attention to the pollutions.

- f_9: comfort level ($a_{n,9}$)
  $a_{n,9}$: The overall comfort level during travels.

- f_{10}: range hour travel ($a_{n,10}$)
  $a_{n,10}$: The range of requested departure.

- f_{11}: price travel ($a_{n,11}$)
  $a_{n,11}$: The willingness to pay evaluates dividing the interval between 0 and 1 in n subinterval.

The M Q-Matrix are therefore built, each one collecting the estimated values of actions available for one characteristic and apply the Q-learning update rule:

$$Q_{i,t+1}(a_t) = (1-\alpha)Q_{i,t}(a_t) + \alpha_t[r_t + \gamma max_{a\in A}Q_{i,t}(a)] \tag{2.48}$$

The objective is then to find the M optimal actions $a_i^*, i = 1, ..., M$, one for each characteristic; the collected optimal actions will form the joint action that will be $\mathbf{a} = (a_i^*)_{i=1,...,M}$.
In order to guarantee a certain degree of exploration, an ϵ-greedy policy is adopted. At each time step t, the best action for each characteristic is chosen with probability 1-ϵ, with $\epsilon \in (0, 1)$, according to the Q-Matrix, i.e.:

$$a_i^* = argmax_{a\in A}Q(a), i = 1, ..., M \tag{2.49}$$

whereas a random action $a \in A_i$ is chosen with probability ϵ. The parameter ϵ has to be tuned to drive the trade-off between exploration of the action space,

with a value for ϵ close to 1, and exploitation of the learned preferences, with a value ϵ close to 0.
The optimal a_i^* will be then used for both : (A) provide a personalized ordered list of solutions and (B) compute a tailored solution (stitched on the preferences and needs of the user).

2.4.5 Adaptive Rank and Results

The estimated traveler model was implemented for acquiring both user needs and expectation, and providing a valuable service to users by exploiting the learned context.
An Adaptive Rank algorithm was developed aimed at proving tailored list of ranked travel solutions to each user.
The user model has been used to model each travel solution TS (i.e. each travel solution in output from a generic journey planner can be converted into features and action of the presented model), the collection of different travel solutions TS_k (k the number of travel solutions) for each query will form the list to be ranked for tailoring the users' experience. The collection of TS_k modeled over the travel model has the following form:

$$TS_k(f_1(a_i), f_2(a_j), ..., f_{11}(a_y) \tag{2.50}$$

The acquired user knowledge in terms of needs and preference, is therefore used to evaluate each travel solution by assigning a value.
As previously described for each action in each individual Q-Matrix (build by following the user model) we associate a weight, and then modeling each travel solution (as described in equation 2.50) we obtain a weight for each action in each feature.

Algorithm 3 Adaptive Rank Algorithm

Initialization

Given

K=Number of solutions

w=w-th Q-Matrix

i, j, y=active actions in each solution

s=1,...,K

Compute

Model the set of solutions TS_K into traveler model

$TS_K(f_1(a_i), f_2(a_j), ..., f_{11}(a_y)$

Take the weights for each travel characteristic from the Q-Matrix as $W_s(TS_K) = Q_w(f_j(a_i))$

Compute the weight for each TS following the equation

$TS_K = \sum W_s$

$\mathbf{W} = W_s$

$\mathbf{W}_{sorted} = sort(\mathbf{W})$

The list of ranked travel solutions, according to the user needs, is therefore obtained by ordering the travel solutions by means of weights and considering a descending order, as described by the **Algorithm 3**.

Now, consider a generic user **A** who begins its first interaction with the system shown in figure 2.12. The learning algorithm such as Single user adaptive learning, designed for acquiring knowledge from the human-machine interaction, doesn't know the user needs, since the user is at the first interaction with the system.

This means that, in case user **A** requests a generic trip, the system provides an initial random list of ranked travel solutions (according to the whole set of alternatives) waiting for a user first choice in order to acquire the first user **A** needs.

Figure 2.13 shows what the system has provided to the user. The set of travel solutions with a tailored random order is shown to the user (referring to top left part of the figure 2.13). In this respect, in the set of travel solutions depicted in figure 2.13, the user **A** selects the third solution, for instance. The selected solution contains paths with FOOT and CAR SHARING. The selection from user allows the system to learn the initial user needs.

Figure 2.14 shows the same solutions, according to the same request, with the difference that the solutions are ordered taking into account the user latest

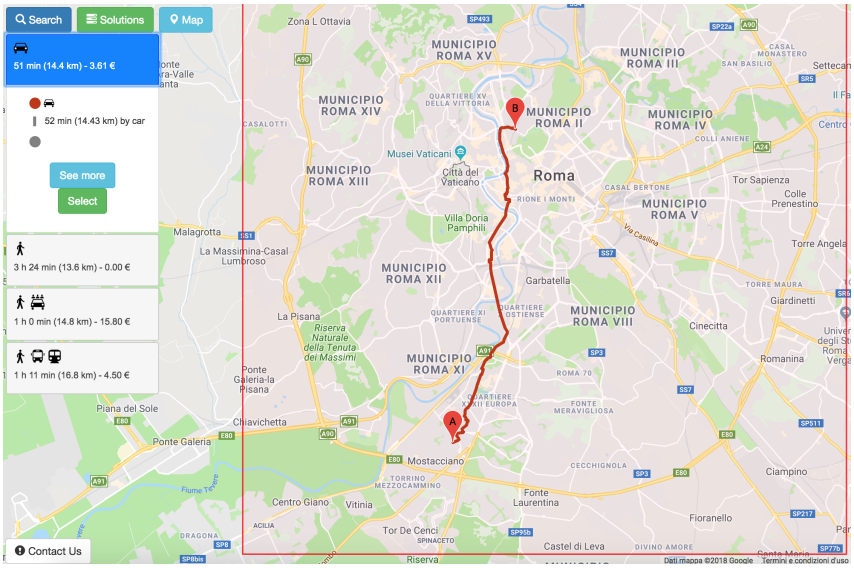

Fig. 2.13 First set of random ranked travel solutions

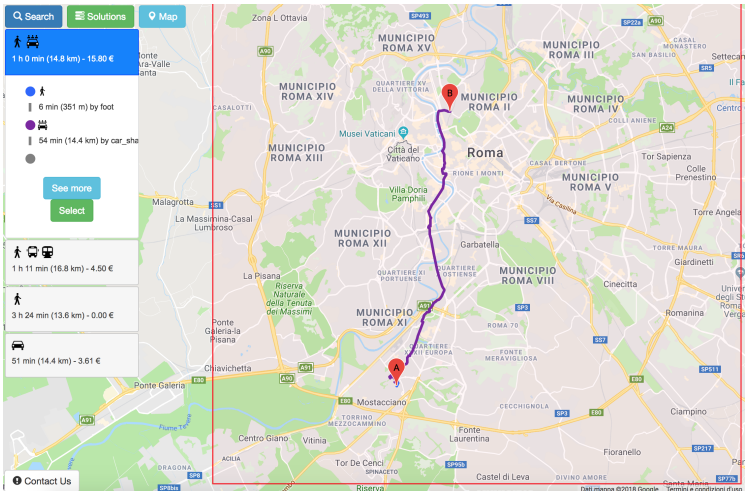

Fig. 2.14 Second set of ranked travel solutions

choice. The adaptive rank algorithm has learned the initial preferences and in this second interaction (see figure 2.14) a more precise order is performed.
In order to figure out the performances in terms of effectiveness of the tailored rank of solutions in Figure 2.15 a new set of solutions is shown and provided by the system to the user **A**. The set of solutions are returned to the user, still according to the acquired needs and preferences, in fact in the first position in 2.15 is still presented the CAR SHARING solution; the other solutions are ranked taking into account the 11 features already presented.
In case the user **A** selects the third solution in the list of 2.15, (METRO BUS) the system acquires again the user preferences and needs. A result of the last choice is presented in Figure 2.16 where the system returns the tailored list of ranked solutions according to the user **A** choices.
The effects of the human-machine interaction have result also in case the query is outside the urban area and covers long range distance as depicted in Figure 2.17 where the solutions are ranked according to the user **A** complete choices. The first solution in 2.17 from OSLO to ROME includes CAR SHARING for traveling in the Rome city as already selected by user **A** in its choices. In particular, the first solution within the whole set of solutions returned by the system contains: (1) CAR as first transport mean to reach the airport, (2) AIRPLANE to reach the Rome airport, (3) TRAIN to link the Rome airport with the Rome city center and (4) CAR SHARING path to reach destination. Furthermore, the fact that the system has learned the user **A** preferences is highlighted by the fifth ordered solution in the list. The fifth solution contains paths not desired by the user **A** taking into account the previous user **A** selections.
The results shown from figures 2.13 to 2.17 try to demonstrate how the system is able to learn preferences from the *human-machine* interaction and project the acquired knowledge for controlling and improving the users' experience while interacting with the application by ordering in a personalized way the solutions for user **A**.

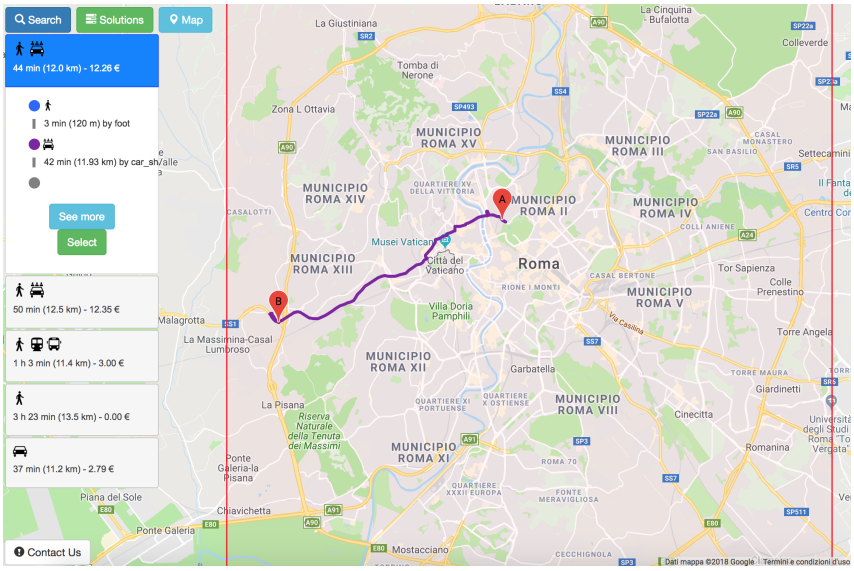

Fig. 2.15 Third set of ranked travel solutions

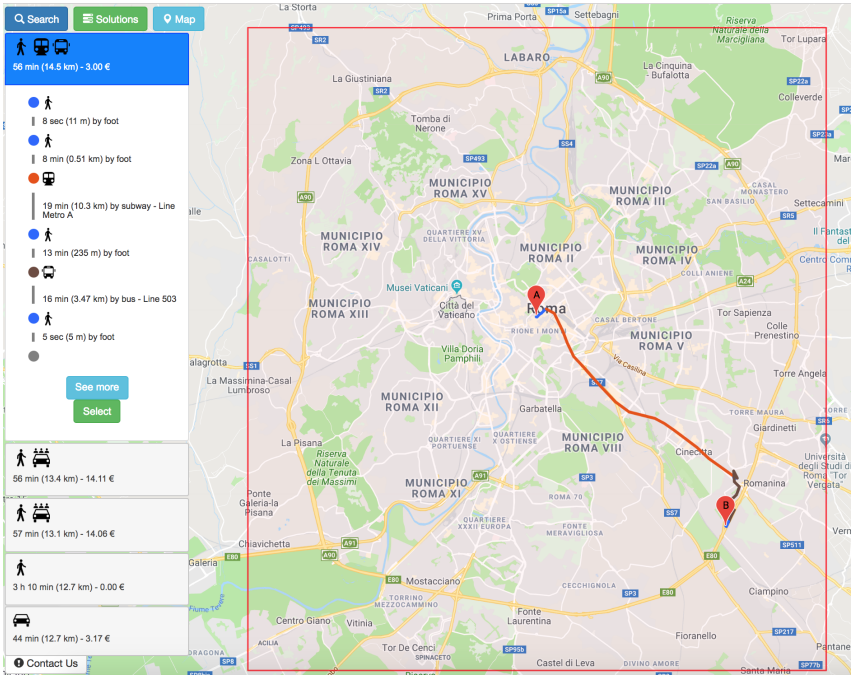

Fig. 2.16 Fourth set of ranked travel solutions

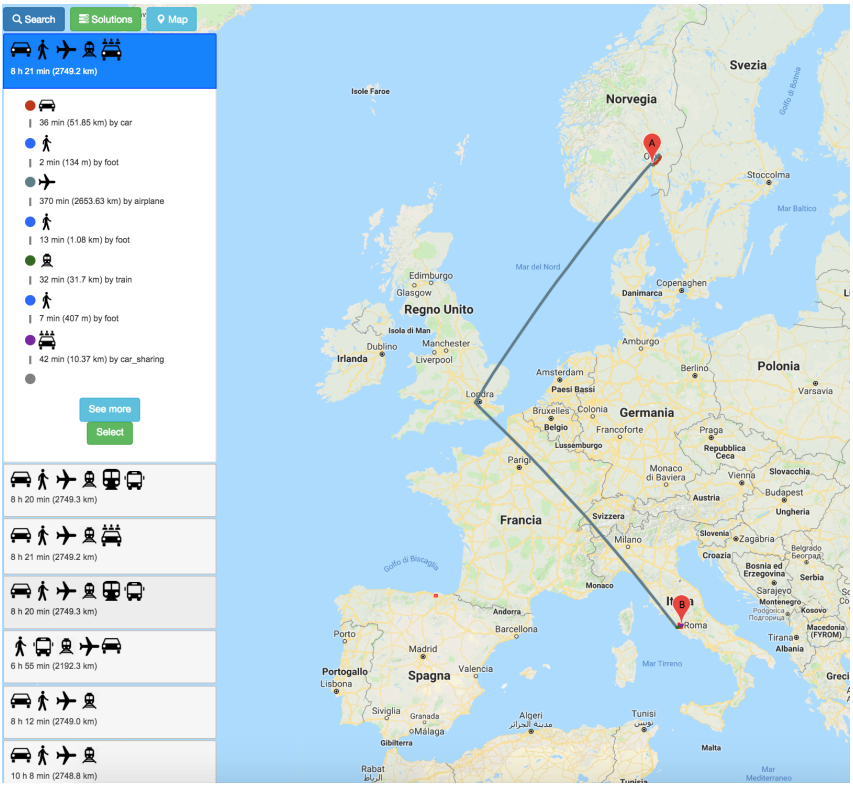

Fig. 2.17 Fifth set of ranked travel solutions

2.4.6 Tailored Solution

As introduced above the adaptive learning system is able to capture the users' behavioral context. The behavioral context can be used to compute the tailored solution by proving individual route parameters to a generic Trip Planner. The generic Trip Planner taking into account the individual route parameters will provide the tailored solution thought to be the most preferred solution. The route parameters RP_i, where $i = 1, ..., 11$ are the optimal actions a_i^* computed according to the equation in 2.49, converted into actions predicates.

$$RP_i = [a_1^*, a_2^*, a^*{}_3, a_4^*, a_5^*, a_6^*, a_7^*, a_8^*, a_9^*, a_{10}^*, a_{11}^*] \tag{2.51}$$

This means that the RP_i contains the actions having the best value for each characteristic according to the equation 2.48. The RP_i are delivered to the Trip Planner that computes the potential "optimal" solution for the user. Figure 2.18 shows the tailored solution according to the user **A** behavioral context; with respect to the previous example where more than one solution is presented, in this case the trip planner returns only one solution considered as the most

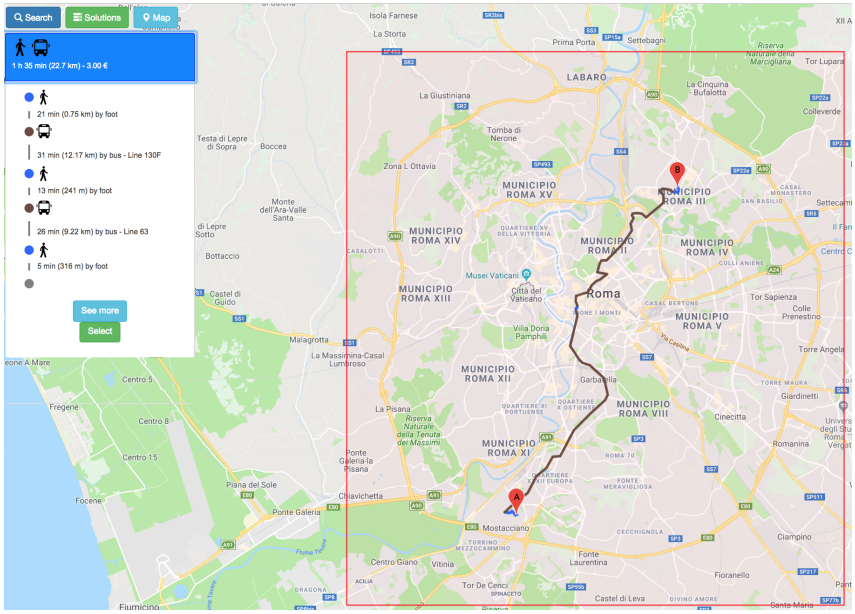

Fig. 2.18 Tailored Solution

desired solution for user **A**.

The suggested RP_i are used by the trip planner as constraints. The trip planner used to perform the simulations reported in this chapter has been developed within the H2020 project BONVOYAGE [2].

[2]BONVOYAGE H2020 http://www.bonvoyage2020.eu/

Chapter 3

Deep Learning based Intelligent Transportation System

3.1 Introduction

In this Chapter, dedicated to the automatic recognition algorithms, I reports the research activities carried out in the field of Artificial Intelligence (AI); more precisely *machine learning* [75](ML) and *Deep Learning methodologies* [37] (DL). The ML and DL have attracted a lot of attention in the recent years for their elevated efficiency in helping human people in several different tasks. Currently, these methodologies are providing great advantages in different fields: physics, computer science, economics, vision analysis, health care, etc.., with a relevant number of applications. The great part of the success is given to the fact that the Artificial Intelligence methodologies are able to bring intelligent capabilities to computers/machine. The human being are able to perform reasoning and make decisions thanks to the acquired experience during their life. There are some hard problems, in terms of computation complexity, in which the human beings could fail either due to the complexity or at least take a lot of time for solving. In general, the ability to solve real world problems that belong to some physical (play guitar) or intellectual (interpret a sentence) become feasible for a human being after a training process.
This is the main concept of the Machine Learning algorithms and related techniques, in fact just like the human beings the Machine Learning algorithms need to be trained before providing a correct decision. Such techniques can be implemented via software, developing an algorithm, that learns exactly what the human being (the developer) has intention to perform. After a period

of training the developed software is able to get decisions with its relative experience.
Moreover, the algorithms will achieve the awareness of a given task after a training process. The training process is mainly based on data analytics, and the Machine Learning algorithms can acquire experience from the past having knowledge about the data which describe for instance past events.
The human being has made-up Machine Learning for solving some complex tasks in which was simpler to develop an algorithm than solving manually the tasks. The ML and DL methodologies are mainly used for pattern recognition, time series prediction, images classification, etc .... However, Deep Learning [37] is a particular Machine Learning technique able to solve the most complex problems.
The Machine Learning algorithms can be divided into three main types:

- **Supervised Learning**
  The Supervised learning algorithms refer to those algorithms trained with a target variable associated to each data entry. Basically, these algorithms generate a function able to map for each input variable a desired output. In the learning process the algorithms take as input a set of different observations and as output a set of different labels representing the outcome of the observations. Examples of Machine learning supervised algorithms are: Logistic Regression [9], Random Forest [11], Decision Tree [95], K-nearest neighbors [58], Support Vector Machine [97], Artificial Neural Network [21], etc ...

- **Unsupervised Learning**
  The Unsupervised Learning algorithms refer to those algorithms that are able to predict or at least estimate a target variable for each data entry. The main different with respect to the Supervised Learning algorithms is that the target/output variables are not known in a priori manner. This means that the data are unclassified and then a correlation among input variables and output variable is unknown. The objective of these algorithms is then a prediction of the output variables and they are used mainly for dividing the data in different groups/clusters. A typical example of Unsupervised learning algorithm is: K-Means [44] and AutoEncoders [98].

- **Reinforcement Learning**
  The Reinforcement Learning algorithm was described in the previous chapter (namely, Chapter 2).

Supervised Learning algorithms are particularly suited for solving two different types of problems:

- **Classification**
  Classification is the problem of identifying the class of a given observation. For example, the observation ca be an image and the class can be considered as what the image represents: a CAR.

- **Regression**
  Regression is the problem of estimating which are the relation among variables involved in a given dataset. These kind of problems can be also called predictors; in fact, the regression algorithms, properly designed for solving regression problems, are able to predict numerical values associated of the given observations. An example of regression algorithms are used for predicting house' pricing (output variables) in the future having knowledge of the prices in the past and the features describing the houses (input variables).

Deep Learning algorithms are particularly used (as already mentioned) for facing with high dimensional dataset and complex tasks in case of both Supervised Learning and Unsupervised Learning.
The machine learning algorithm, previously cited, are well used for simple and normal data analysis. In case of a huge amount of data the simple machine learning algorithms are not enough powerful to distinguish among highly heterogeneous dataset. In this regard, one of the main authors and follower of Deep Learning methodologies, namely Ian Goodfellow, in his book says [37]:

"Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representation computed in terms of less abstract ones".

The first part of this chapter is devoted to introduce Deep Learning techniques, architecture and algorithms. The second part of the following chapter

is devoted to introduce the transportation mode recognition problem and how I have solved the problem with Deep Learning methodologies.
The transportation mode recognition will be used for monitoring the traffic flows enabling the possibility to control traffic lights in Vehicular Networks as presented at the end of this chapter.

3.2 Deep Learning

The main advantages brought by Deep Learning methodologies cover the ability to (i) analyze a huge amount of heterogeneous data; (ii) obtain high performances in terms of accuracy (a measure of error computed as the different between the estimated and the target value) for classification or prediction; (iii) create the correlation among sequential data; (iv) high level of flexibility in complex tasks as for example self-driving car; (v) automatic identification of relevant and/or new features in a dataset.
In the presence of great advantages in the field of data science, Deep Learning suffers from several different disadvantages. The main disadvantages are strictly related to the dimension of the data that will have to be analyzed. In fact, Deep Learning offers its great power only in case of huge amount of data, while losing accuracy and efficiency in case of small data collection. The second, but not less important, disadvantage is that Deep Learning algorithms need a long time to be trained and at the same time they need a great computational power to run. Anyway, nowadays the human people have figure out the power of analyzing data and they are collecting as much data as possible in several different fields.
To present a possible comparison, in terms of how Deep Learning is more powerful than Machine Learning, we can introduce a clear example:
We would like to develop a prediction system able to ($*$) solve a classification problem in charge of identifying if a image contains a Car or a helicopter, and ($**$) solve a clustering problem able to associate similar things to the same group.
For solving the problem in ($*$) with classical machine learning algorithms the data scientist has to identify a method for extracting image features and then define a function for capturing the main differences between images represented with a features selection (e.g. the helices of the helicopter). Using a Deep Learning methodology particularly suited for image classification such as Deep

Convolutional Neural Network (DCNN), the data scientist has to only identify the DCNN parameters that better provide the classification (the parameters identification and more details about DCNN will be presented later).
For solving the problem in $(**)$ we need to use a clustering machine learning algorithm such as k-Means. Since the clustering algorithm has no knowledge about labels it has to autonomously (without a supervision) divide the level of belonging of the different images. The tasks can not be addressed by using classical machine learning [57] since K-Means is not able to classify images whose features are non-linear, due to the fact that K-Means algorithm is a linear model.
Deep Learning techniques and algorithms are based on Artificial Neural Networks (ANN) [21]; in particular, Deep Learning refers to such neural networks having multiple hidden layers. From the hidden layers perspective, they are composed by several different artificial neurons or perceptrons.

3.2.1 Perceptron

The Perceptron or Neuron can be considered as the fundamental unit of artificial neural networks, it is inspired by the biological brain where the synapses provide biological signals in mammals. In order to respond to certain stimuli similar to the brain, the perceptron implements activation function. The activation functions are also biologically inspired to the action potential (nerve impulse).

The Perceptron is a linear discriminant model and it is able to process input values (also called features) and provides in output values according to the input.
The Perceptron algorithm [82] was created by Frank Rosenblatt in 1957. It was designed to be an algorithm for binary classification.
The perceptron, showed in Figure 3.1 takes as input a vector of real-values $\mathbf{x} \in \mathbb{R}^n$, that is transformed with a nonlinear transformation in the form of a vector $\phi(\mathbf{x})$ [8]; the nonlinear transformation is used to build a generalized linear model having the form:

$$y(\mathbf{x}) = f(\mathbf{w}^T \phi(\mathbf{x})) \tag{3.1}$$

In the generalized linear model presented in the equation 3.1 the $f(*)$ represents the linear activation function or step function in the perceptron having the

following form:

$$f(a) = \begin{cases} +1, & a \geq 0 \\ 0, & a < 0 \end{cases} \tag{3.2}$$

With the equation in 3.2 is simple to understand the reason why the perceptron algorithm can be consider as a binary classifier, in fact it is able to determine the relative class C_1 or C_2 for the given input $\mathbf{x}$. This means that the perceptron maps each input into a class which correspond to the value in the activation function. For each input (observation) (x_i), where $i = 1, ..., N$ in the input vector $\mathbf{x}$ are associated a set of weight w_j, where $j = 1, ..., N-1$, the $w_0 = b$ where b is a *bias* threshold. The w_i is the value for connecting the relative input value x_i and the output (classification). The key concept in perceptron

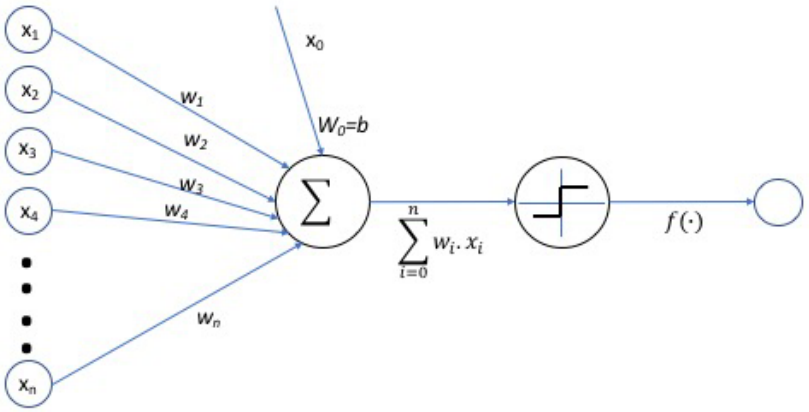

Fig. 3.1 Single Unit - Perceptron - artificial neuron

algorithm is that the algorithms attempts to learn w_i parameters for the correct classification of the inputs. To learn w_i, the algorithm used are perceptron rule and delta rules [75]. The convergence of the perceptron rule algorithms happens in case of linearly separable dataset and the percetron rule fails in case non linearly separable data in learning dataset. Let's assume a dataset $\mathbf{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ..., (x_n, y_n)\}$ which contains for each input x_i (known observation) the corresponding output y_i (target for the given input). From the perceptron perspective, the target values are binary (0,1) but in case of more complex neural networks they could be multi categorical values (classification problems) or real values (regression problems).

As already introduced, the perceptron is the fundamental unit of an Artificial Neural Network (ANN). The perceptron has a very limited capacity, since it

is able to learn only linearly separable examples. In fact, in case of more complex problems with non linearly separable examples (inputs), more than one perceptron/neuron can be interconnected with other neurons creating the ANN that has the capabilities of approximating continuous functions with the assumption of the activation function.

3.2.2 Artificial Neural Networks

The Artificial Neural Networks are designed to have more connected units (perceptron or artificial neurons), with interlinked inputs, outputs and hidden nodes.
In artificial neural networks, each connected artificial neuron provides a signal to the other connected neurons creating an overall connected system. In the artificial neural networks the signals are real numbers computed by each perceptron, composing the artificial neural network, as described in the previous paragraph.
As already introduced the learning process in perceptrons is enabled from the **weights** w_i describing the perceptrons. The systems with multiple connected perceptrons, characterizing the artificial neural networks, have a real relevant number of weights w_i. The **inputs** for the artificial neural networks are the features which describe the context to be learned by the artificial neural networks, and the output is computed with particular non-linear activation function.
In case of non-linearly separable dataset the weights can be learned by back-propagating the loss using the back-propagation method [36]. The loss is computed with loss function able to measure the difference between the predicted output $\tilde{y_i}$ and the real output y_i. Dependently by the nature of the dataset and the problems that would like to be solved by using neural networks there are different back-propagation optimization algorithms (gradient descent, stochastic gradient descent, adam, etc...), loss functions (Binary cross Entropy, Margin Classifier, Soft Margin Classifier, Absolute Loss function, mean Square Loss Function, etc...) and activation functions (e.g., Binary Step, Logistic/Softmax, TanH, ArcTanh, ReLu, Sinc, Gaussian, etc...).

By increasing the connections and the number of units interconnected in the ANN, it is then possible to assign to the ANN very complex tasks thanks to the number of weights that can be adjusted and that then can particularize the learning process.

Typically the Artificial Neural Networks are composed by neurons for the inputs, neurons computing the outputs, and some other connected layers (each layer is characterized by a number of neurons) that can be called hidden layers (hidden nodes) since they are typically located between the inputs and outputs. Figure 3.3 shows an example of an ANN with only one hidden layer.
When the ANNs have many hidden layers they are called *Deep Artificial Neural Networks*. There exists several different Deep architecture each one designed for solving different problems. An example of common Deep Artificial Neural Network is the Deep FeedForward neural network [30], depicted in Figure 3.2 where the connection between the different nodes and layers composing the networks doesn't have cycles and the information are propagated along the network in one direction, forward.

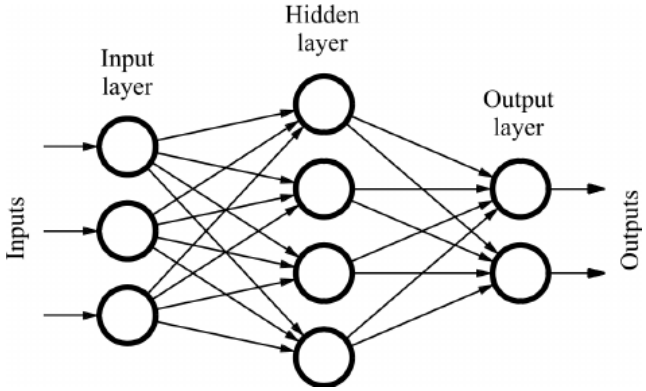

Fig. 3.2 Artificial Neural Network architecture [21]

The Deep Feedforward neural networks are used for supervised learning problems considering both Classification and Regression problems.

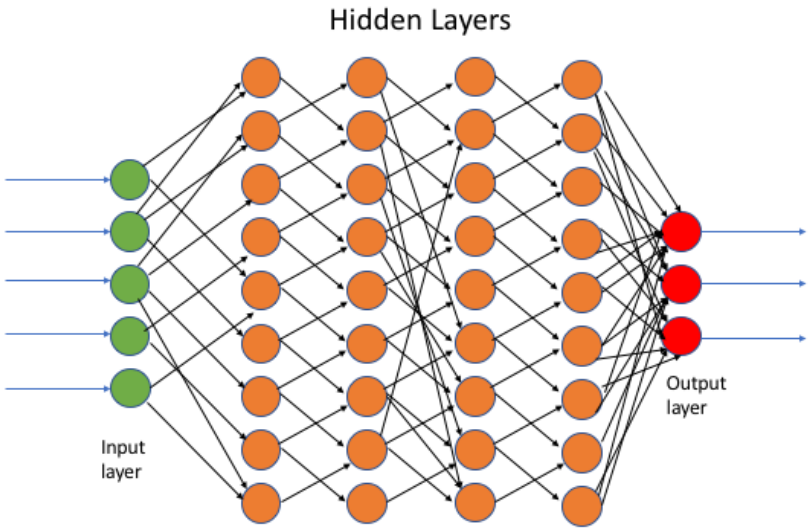

Fig. 3.3 Deep FeedForward Neural Network architecture

3.2.3 Deep Recurrent Neural Networks

Deep Recurrent Neural Networks (RNN) are a particulate type of Deep Architecture properly design for sequential inputs (temporal data). They are particularly suited for prediction problems, where given a sequential data inputs they predict the next value.
Differently from the Deep FeedForward these type of networks form cycle among connected neurons. This type of Networks can be used in various fields as for example: machine translation, time series predictions, sentiment analysis, etc... .
Differently from other Deep Networks the Recurrent Networks attempt to maintain association between past and recent events, in contrast with other Deep Architecture that don't maintain memory from the past information. In order to associate past and recent event it implements a transition weight in order to get the memory and provide information among times.

Simple RNN

In Deep FeedForward Neural networks all the inputs are independent, in the sense that they have not any relation each other; while in case of time series data, where each data in the future has a relation in the past, the assumption of independence among input data is broken. The Simple RNNs, depicted in

Figure 3.4, have a sort of memory in the hidden state which creates and takes into account past events written in the time series examples.
Given a sequence of inputs denoted with $x_1, x_2, x_3, ..., x_n$, in classification or regression problems correspond an output sequence in the form $y_1, y_2, y_3, ...y_n$, the Recurrent Neural Network computes the hidden vector sequence $h = (h_1, ..., h_T)$ by repeating the equation in 3.3 and 3.4 for $t = 1, ...T$. In Recurrent Neural Network we have to pay attention between what the network should provide in output $y_1, y_2, y_3, ...y_n$ and what it actually provides $\tilde{y_1}, \tilde{y_2}, \tilde{y_3}, ..., \tilde{y_n}$. The Recurrent can provide either an output for each input or an output for a set inputs.

$$h_t = tanh(Ux_t + Wh_{t-1} + b) \tag{3.3}$$

$$\tilde{y_t} = softmax(Vh_t + c) \tag{3.4}$$

The equations in 3.3 and 3.4 describe the behaviour of the Recurrent Networks. In particular, the RNN computes the h_t for a given input x_t and taking into account the h_{t-1}; the output of the RNN is computed considering the h_t.
Now, the weights associated to the input are represented as U, while the weights of the previous hidden state are in W while computing the h_t. V in equation 3.4 are the weights of the h_t computing the output. The Simple RNN uses the $tanh$ to implement the recurrence.

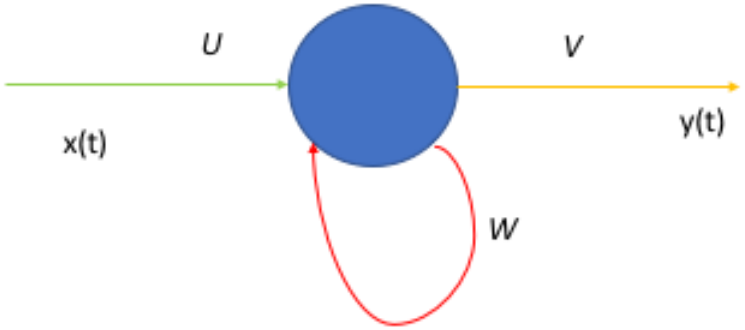

Fig. 3.4 Simple RNN architecture

Long Short Term Memory

A type of RNN model is the Long Short Term Memory (LSTM) [46], particularly

suited in case of long term dependency between time series examples.
The LSTM model is similar to the Simple RNN in terms of how they implement the recurrence, but instead of a single *tanh* (see equation 3.3) in LSTM there are 4 layers devoted to each specific task for implementing the recurrence.
In Figure 3.5 the function of a LSTM hidden layer is depicted: $c(t)$ represent

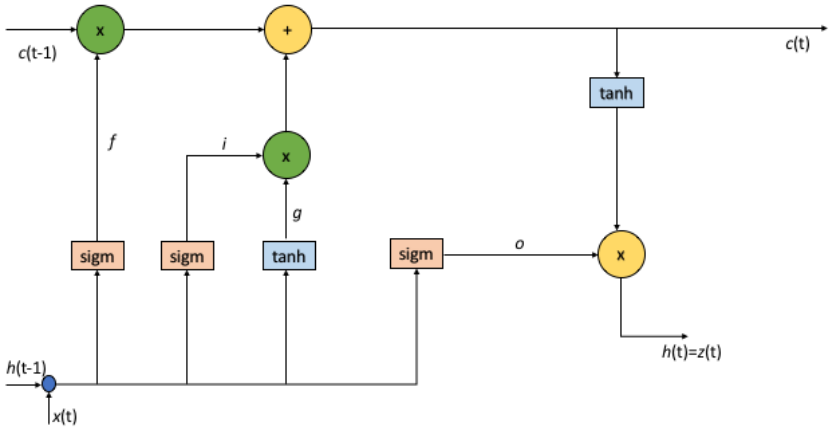

Fig. 3.5 LSTM architecture

the state at time t representing the internal memory, while $h(t)$ represent the hidden state; the gates i, f, g, o are parameters to be learned during the training process and then overcoming the vanishing gradient problem [26].

$$i = \sigma(W_i h_{t-1} + U_i x_t) \tag{3.5}$$

$$f = \sigma(W_f h_{t-1} + U_f x_t) \tag{3.6}$$

$$o = \sigma(W_o h_{t-1} + U_o x_t) \tag{3.7}$$

$$g = tanh(W_g h_{t-1} + U_g x_t) \tag{3.8}$$

$$c_t = (c_{t-1} \otimes f) \oplus (g \otimes i) \tag{3.9}$$

$$h_t = tanh(c_t) \otimes o \tag{3.10}$$

The above equations represents how the gates i, f, g, o influence the hidden state.
Now $h_t \in \mathbb{R}^N$ the LSTM includes the forget gate $f_t \in \mathbb{R}^N$, the input gate $i_t \in \mathbb{R}^N$, an output gate $o_t \in \mathbb{R}^N$ and an input modulation gate $g_t \in \mathbb{R}^N$.
The LSTM are designed to learn from long-term dependency among data, and the implementation of the presented gates allows the learning for long-term

dependency.

Bidirectional LSTM

Another example of interesting RNN model is Bidirectional RNN [85] that also can be transformed in Bidirectional LSTM [39] by combining Bidirectional RNN with LSTM model.

A limit of standard RNN is that they learn from the past but not from the future context during the learning process [38], this means that the RNN has not knowledge of future input from the current state and the output depends on what the network has learned only from the past, this restriction is exceeded by Bidirection RNN (BRNN) designed to reach future input from the current state. The BRNN needs to have two networks or hidden layers (a) backward layer and the (b) forward layer, where the layer in (a) has access to the past information from the current state, while the layer in (b) has access to the future information, from the current state, if present in the dataset. The output in the BRNN will be effected by both past and future information from the current state.

Since the BRNN has two hidden layers it computes two hidden states h, the former for the backward layer $h_{t,back}$ and the latter for the forward layer $h_{t,for}$ for $t = 1, ..., T$. The equations from 3.11 to 3.13 describe the BRNN. In this respect, $h_{t,back}$ and $h_{t,for}$ represent the hidden states, backward and forward respectively, computed from the input at x_t and the previous hidden state $h_{back}(t-1)$ and $b_{for}(t-1)$. The W and U are the weights associated to previous hidden state for the input during the computation of the $h_{t,for}$ and $h_{t,back}$ considered as current hidden state.

$$h_{t,for} = tanh(U_{for}x_t + W_{for}h_{t+1} + b_{for}) \tag{3.11}$$

$$h_{t,back} = tanh(U_{back}x_t + W_{back}h_{t+1} + b_{back}) \tag{3.12}$$

$$\tilde{y} = softmax(V_{back}h_{t,back} + V_{for}h_{t,for} + c) \tag{3.13}$$

The BRNN can be extended into Bidirectional LSTM [38] for long-term evolution and Figure 3.6 shows the Deep Bidirectional LSTM architecture.

Now, let's us provide a concrete example to present the main difference between the LSTM and Bidirectional LSTM and how the Bidirectional LSTM provides a valuable improvement in predictions.

Now, we can consider a word generation problem where the sentence is charac-

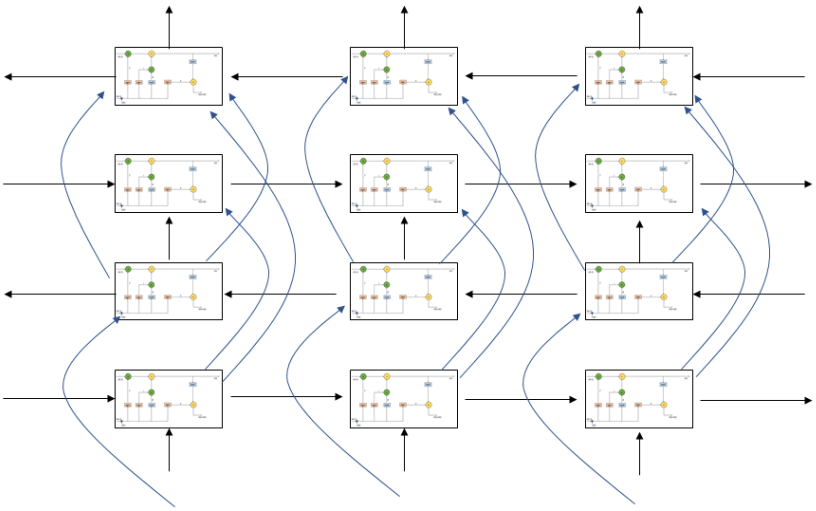

Fig. 3.6 BLSTM architecture

terized by the following expression:

I am a . . . . . player.

In the LSTM to estimate the words after the empty space in the sentence (i.e., ". . . . . ") only the words that occur before the empty space will be taken into account and then only '*I am a*', the LSTM will predict the words taking into account the sentence learned during the training process. In this case for example the LSTM could predict "student".

In the Bidirectional LSTM since the networks shall take into account the past events, that is *I am a* and the events after the empty space, that is *player*, based on the experience acquired during the training process the Bidirectional LSTM, potentially will predict "soccer", that it is evident to be the most appropriate word in this example.

3.2.4 Deep Convolutional Neural Networks

The Deep Convolutional Neural Networks (DCNET) [2] are a type of Deep Architecture designed to perform in two-dimensional data for classification of images or movies, and, indeed, they can be used for several different types of applications involving classification (i.e., recommender systems, behavior recognition, speech recognition, etc . . .).

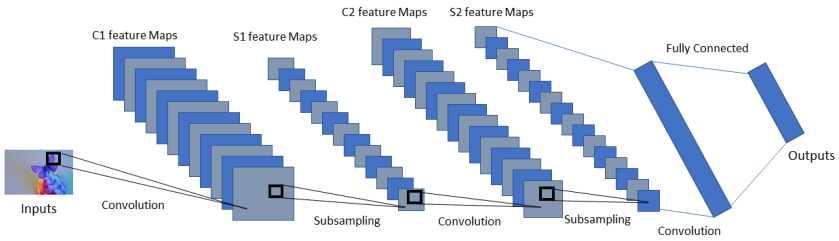

Fig. 3.7 Deep Convolutional Neural Network Architecture

The standard Deep Networks, for instance Deep FeedForward Neural Network are not able to scale in case of high dimensional images, on the contrary the DCNET are properly designed to perform well also in case of high dimensional images.

The DCNET are made up focusing on the brain of animal visual cortex and in particular on the receptive fields [24] and technically inspired by a previous work on *time-delay neural network* [2] which consists in sharing the networks weights in a temporal dimension [99], that fosters the possibility to reduce the computations.

The main advantage in DCNET is that the data input vectors (e.g., images, videos, documents) don't need to be preprocessed for features extraction [68]. The DCNET are characterized by the convolution operations, in Figure 3.7 the DCNET architecture is presented having three main layers, namely the *convolution layer*, the *subsampling* or *pooling layer* and the *fully connected* layer. The black boxes in Figure 3.7 represent the *filter* that is used to perform the convolutions among outputs from layers. The sub-sampling layers perform the pooling [62] operation that consist in selecting the best valuable value into each filter matrix between two convolutional layers that enable the dimensionality reduction. The fully connected layer is typically used at the end of the convolutional operation in order to obtain the classification/recognition.

In order to go more in depth with the representation of how DCNET works,

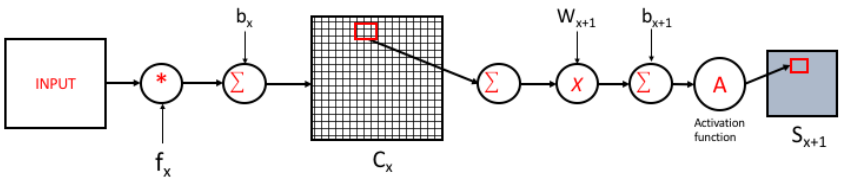

Fig. 3.8 Convolutional Operation

we can refer to an example shown in Figure 3.8 [2]. In DCENT the convo-

lution steps involves the convolution of an input (for instance an image if the convolution is in its first stage or feature maps in case the convolution is involved in further steps) with the filter f_x and a bias b_x in order to compute the convolution layer C_x. The subsampling process concerns a set of different operation (in Figure 3.8 are those operation between the C_x and S_{x+1}) to obtain a feature map S_{x+1} by using an activation function properly designed for the scope.
One of the most important characteristic in DCNET are the shared weights properties, in fact the learned weights during the learning process are shared for the whole input in each input fashion.
The filter has dimension $m \times n$ and it is applied to the whole input moving across the input, or in case of second convolution layer moving into feature maps. Having a input size of $X \times X$ with a filter $m \times n$ the feature map size created by using the convolution operation is:

$$\frac{X - p}{s + 1} \tag{3.14}$$

where s is the dimension of the *stride*. The stride determine how many sequences of the input have to be convolved. Bigger is the stride, bigger is the size of feature maps.

3.2.5 Deep AutoEncoders

The Deep AutoEncoders (DAE) [98] are a particular type of Deep Architecture (see Figure 3.9) designed for unsupervised learning. The DAE are particularly suited to synthesize huge amount of data extracting encoded features from the input data in an unsupervised manner. The encoded features representing the dataset could be considered of a *model* of the data.

The operating principle covers the possibility to compress the input data (i.e., the dataset) x into a numerical encoded representation, then the encoded representation is re-built obtaining in output $x^{'}$ which has the same representation of the given input.
In DAE the input data $\mathbf{x}$ intuitively represents the dataset.

$$\mathbf{x} \in [0, 1]^d \tag{3.15}$$

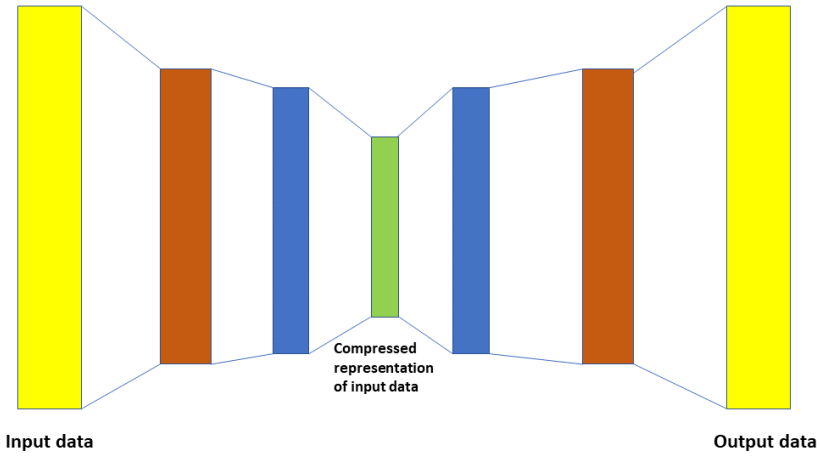

Fig. 3.9 Deep AutoEncoders Architecture

The DAE maps the input data in 3.15 into a hidden representation of the form 3.16 [98]

$$\mathbf{y} \in [0,1]^{\tilde{d}} \tag{3.16}$$

by using a deterministic mapping where $\theta = \{\mathbf{W}, \mathbf{b}\}$ and the sigmoid $s(x) = 1/(1+e^{-x})$

$$\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{3.17}$$

the matrix of weights $\mathbf{W}$ in 3.17 is

$$\mathbf{W} = d \times \tilde{d} \tag{3.18}$$

and $\mathbf{b}$ represents the bias vector. The $\mathbf{y}$ represents the compressed or encoded description of the input vector in $\mathbf{x}$, and it is used to re-build the input vector having the form $\mathbf{z}$

$$\mathbf{z} \in [0,1]^{d} \tag{3.19}$$

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}) \tag{3.20}$$

In this case the $\theta' = \{W', b'\}$ represents the weights and bias for the rebuilding step.
Therefore, the encoded representation of $\mathbf{x}$ is $\mathbf{y}$ and $\mathbf{z}$ is the rebuilt output.
As explained in [98] the θ and θ' are optimized to minimize the error during the rebuilding model as:

$$\theta, \theta' = argmin_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}^i, \mathbf{z}^i) \tag{3.21}$$

where $\mathbf{x}^i$ and $\mathbf{z}^i$ are the input and output data. By using the equation 3.20 in 3.21 we can obtain:

$$\theta, \theta^{'} = argmin_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}^i, g_{\theta'}(f_\theta(\mathbf{x}^{'}))) \tag{3.22}$$

As already explained in the previous sections to compute the optimization of weights in Deep Neural Networks the loss function is used for the case of DAE, where the *input data is numerical*, in 3.20 and 3.21 the L represents the loss function of the form:

$$L(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}||^2 \tag{3.23}$$

In case the input data x, that must be rebuilt to achieve the compression $\mathbf{y}$, are not numerical but are vectors, for instance, the loss function L that can be used is the cross-entropy as:

$$L_H = -\sum_{k=1}^{d} \mathbf{x}_k log(\mathbf{z}_k) + (1 - \mathbf{x}_k) log(1 - \mathbf{z}_k) \tag{3.24}$$

d in 3.24 is the dimension of the input vectors.
The DAE are in some sense very similar to the dimensionality input data reduction process called Principal Component Analysis (PCA) [109].

3.3 Transportation Mode Recognition Problem

The Transportation Mode Recognition (TMR) [73] problem has attracted much attention in the recent years thanks to its potential use in various fields. The TMR consists of identifying automatically the transportation mode with which users are moving, and this information can be provided to the public transportation provider, National department of infrastructure, as well as to traffic management departure. According to the information acquired with the TMR solution the public transport provider could improve/reduce the frequency of their public transports according to a data based motion flows in cities, the National department of infrastructure could improve/reduce the size of the streets as well as increase or re-design data based special preferential lanes, and the traffic management could control the traffic with traffic lights.
The real-time identification of transportation modes behavior of humans per-

mits to have information about traffic flows to make real-time journey planners being aware about traffic status in smart cities, estimate the CO_2 emission [72], and determine the motion patterns about users in cities. In order to acquire useful information for detecting the transportation mode the use of smartphone sensors is essential. The modern smartphone are equipped with several different sensors as GPS system, accelerometer, gyroscope, magnetometer, etc. By collecting the data provided by the smartphone sensors, and processing those data, it enable the possibility to realize a complex motion model system.
Much literature has already introduced and investigated this problem bringing innovative solutions. For example, in [87] the authors have investigated the possibility to use accelerometers information to determine motion patterns implementing several different classifiers (e.g. Support Vector Machine, AdaBoost, random forest and decision tree); the accelerometers data were pre-processed and a features extraction process was set up for enabling recognition for 4 different transport modalities, i.e., *Walk, Bicycle, Car, Train* with a time window of data collection for each recognition of about **5 seconds** achieving high performances in terms of mean accuracy in recognition of about 99.8 percent. In [72] the authors have used the Fast Fourier Transform (FFT) to extract features from sensors data achieving performances in terms of accuracy in pattern recognition about 82.14 percent for 8 different transport modalities i.e., *Bus, Metro, Walk, Bicycle, Train, Car, Still, Motorbike* and using a time window data collection for enabling the recognition of about **5.12 seconds**. In [45] the authors have collected data by using three different smartphones and implementing different classification algorithms and introducing an innovative technique for estimating the gravity component [45] reaching a mean accuracy over 80 percent for 7 different transport modalities: *Walk*, *Bus*, *Still*, *Train*, *Metro*, *Car* and *Tram*. For enabling the recognition, close to real-time, with the mentioned transport modalities they have overlapped two time windows of data collection of **1.2 seconds** and then used a final time window of data collection for enabling the recognition of **2.4 seconds**. In [101] the authors by using a time window of data collection of about *8 seconds* have a mean accuracy for 6 transportation modalities i.e., *Still, Walk, Bus, Train Metro and Tram* of about 80 percent.
In the real-world where the motion recognition is on-going (basically when it works at full capacity) each user's smartphone sends its motion data information for enabling the motion recognition. To provide those motion information the

smartphones consume their own power. In fact, less is the time window of data collection and less will be the power consumed by each smartphone. A small portion of data collection is then essential in order to save the battery of each smartphone and it enables a new challenge that consist of detecting with high accuracy the motion pattern, despite the data collected are few.
While the idea of detecting in real time the transportation mode is not really innovative I have conducted research activities to implement a real-time transportation mode recognition designing an innovative Deep Learning algorithm able to solve the problem with time windows of data collection under the second.
A real-time transportation mode detection approach also permits to compute in real-time the user motion flow and/or traffic status enabling the real-time traffic control.
In the following paragraphs I will present the designed detection mode system for 7 different transportation modalities i.e., *Car, Walk, Motorbike, Tram, Subway Bus, and Still* with two Deep Learning approaches, namely **features extraction based** and **raw data based approaches**, able to solve the TMR problem. In more details, I will present a feature extraction based approach, where the data from sensors were pre-processed to create the inputs for Deep Recurrent Neural Networks algorithms, and a raw data based approach with an innovative Deep Convolutional Architecture having great performances in terms of detection accuracy and enabling the real-time detection.

3.3.1 Problem Statement

The research activities performed to solve the Transportation Mode detection problem involve several different tasks. Figure 3.10 shows the complete detection system designed from scratch. In more details, for the two mentioned approaches the detection system was divided in two steps. In the Training Step (first step) an Android Based APP was developed (Data Collection APP) able to acquire data from sensors (GPS, Accelerometer, Magnetometer and Gyroscope), creating a set of trip datalogs (a small portion of the trip datalogs are also available on-line) which contains data from smartphones sensors acquired during different trips for the whole set of transport modalities i.e., *Car, Walk, Motorbike, Tram, Subway Bus, and Still*. The raw data in trip datalogs are used for both (i) pre-processing the data and extracting features for training the Deep Recurrent Neural Network and (ii) directly training the

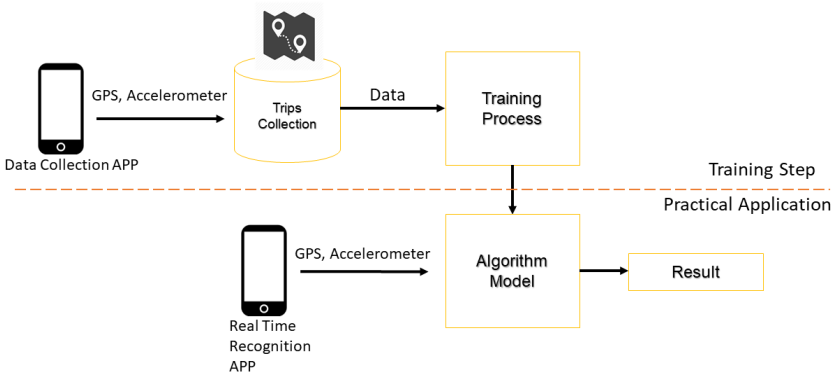

Fig. 3.10 Transportation Mode Detection

Deep Convolutional Neural Network.
After the training process (Practical Application) an *Algorithm Model* in obtained containing *executable model* able to perform the motion pattern recognition with new data coming from the *Real Time Recognition APP*.

3.3.2 Data Collection

As already introduced an Android APP was developed to (i) read and acquire sensor data information, for the purposes of training algorithms, and (ii) store those information in datalogs. The sensor information are referred to Latitude and Longitude from GPS system, Accelerometer for the three axes (x-y-z), Gyroscope for the angular velocity on 3 axes (x-y-z) and Magnetometer on 3 axes (x-y-z). Android permits also to acquire the Current Speed by computing the variation of Latitude and Longitude during the smartphone movement.
These sensor data information are stored in .csv file with a data structure having the form:
{day, month, year, second, minute, hour, signal strength, speed, latitude, longitude, x-acceleration, y-acceleration, z-acceleration, x-gyroscope, y-gyroscope, z-gyroscope, x-magnetometer, y-magnetometer, z-magnetometer}
Figure 3.11 shows the resultant acceleration computed as the root of the relative acceleration over the three axes squared, as in equation 3.25. Through figure 3.11 it is clear that the resultant acceleration differs depending on the transportation modality. In fact each transportation modality has its own

amplitude and dominant frequency of the time-varying acceleration.

$$a_r = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.25}$$

The sensor information are acquired with a sampling rate of 50 Hertz, and

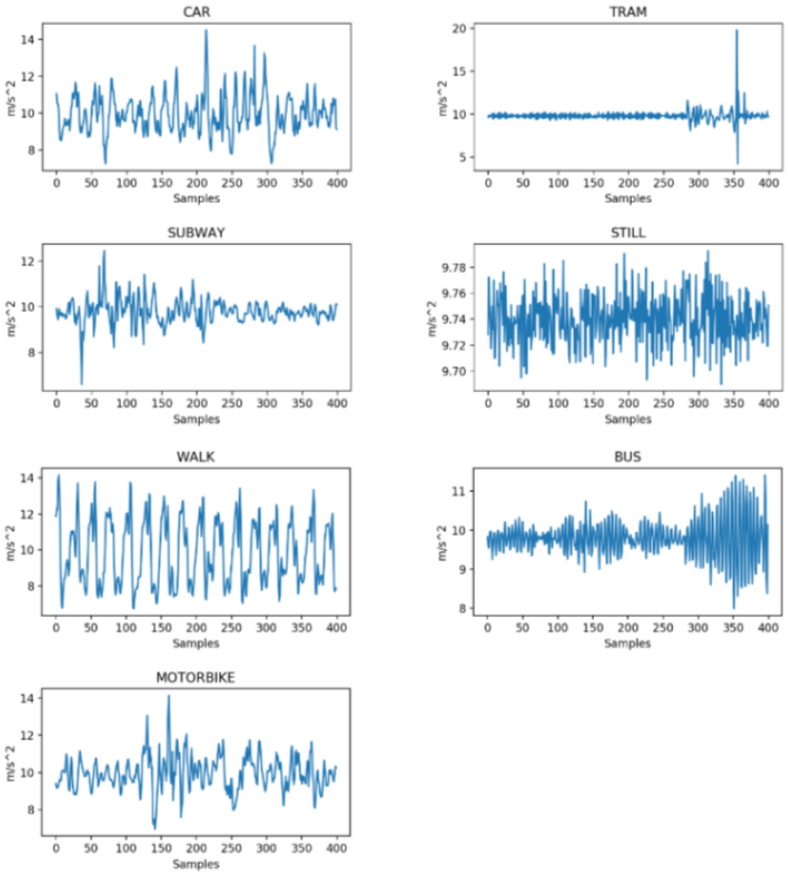

Fig. 3.11 Resultant acceleration over 8seconds

each datalog contains 400 rows (8 seconds) with the data structure presented above.

In totally 140 hours of recording trips have been collected for the whole set of the seven transportation modalities.

The data collected are used to train the algorithms (they will be presented in the following paragraphs) with two approaches. In the first approach the raw data collected are pre-processed and a features extraction phase is performed for training purposes. In the second approach the raw collected data are given in input directly to the Deep Learning algorithm. The two approaches will be compared in terms of:

(i) Performances: accuracy in the classification
(ii) Computational Effort: how many resources are required to perform the classification
(iii) Time Window: how long is the time window of data collection to achieve acceptable performances.

3.3.3 Features Extraction approach

The first approach consists in a two steps solutions. At the beginning (see Figure 3.12), after the data are collected the features are extracted from the raw data stored in the datalogs; the second step in figure 3.13 is characterized

Fig. 3.12 Features Extraction based approach

by a feature-based classification, where the algorithm model classifies the new data coming from sensors of a real time recognition APP once the raw data are

Fig. 3.13 Features Extraction based classification

therefore pre-processed via features extraction for the classification purposes. The extracted features are summarized in the table below.

INVOLVED SENSORS	EXTRACTED FEATURES
GPS/Speed	Mean speed, Max speed, Min speed, Speed std, Speed 5th Percentile, Speed 95th Percentile
Acceleration	Mean acc, Acc std, Max acc, Min acc, Acc 5th Percentile, Acc 95th Percentile, Acc kurtosis, Acc skewness, Acc components at {1,2,...,15}Hz, Acc increase avg, Acc decrease avg, Acc increase max, Acc decrease max
Gyroscope	Mean ω, ω std, ω kurtosis, ω skewness

The extracted features from the three different sensors are 37 taking 50 samples of the sensors data. Each features vector in input to the Deep Recurrent Neural Network is composed by 74 columns computed by aggregating

two slot of the 37 values. Since the sampling time of our Collection Data APP is 50 Hertz the data collection is about 1 second for each slot and then 2 second of data aggregation for each features vector. The other approaches features extraction based presented in literature use more than 2 seconds of data recording for computing the features. The main advantage in reducing the data collection in features based approach is that the computational time to compute features is reduced enabling a faster classification in real environments.

3.3.4 Raw Data approach

The second classification approach such as Deep Convolutional Neural Networks, are able to provide classification without a features extraction process by inferring correlation directly among the raw data. By exploiting the main

Fig. 3.14 Raw Data approach

functional characteristics of Convolutional Neural Network able to automatically create features maps (see chapter 3.2.4), their generalization properties try to deduce autonomously the performing "features" that better approximate the whole dataset (the whole dataset is composed by all the data records for all transportation modalities). Therefore, whereas in the first features

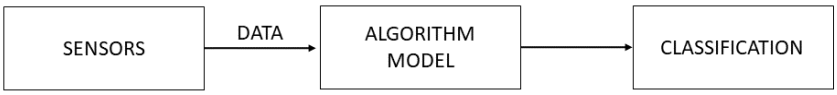

Fig. 3.15 Raw Data Classification

based approach we give the 74 values in input to the algorithm the Deep Convolutional Neural Networks considered in this second approach work will be fed directly with the raw data coming from the sensors. In this case each raw data vector is composed by 4 components represented by the *Speed, x-Acceleration, y-Acceleration and z-Acceleration.* Even in this case the algorithm is trained as depicted in Figure 3.14 and the algorithm model is used in the real environments (see Figure 3.15).

The Deep Convolutional NN in this approach was trained with vector of 64×4 (64 rows and 4 columns) corresponding to 1.28 seconds of data at 50 Hertz. With respect to the first approach introduced in the paragraph above this solution has two main advantages. In fact, it works with a smaller time windows and it doesn't need computation effort for computing the pre-processing step. The mentioned advantages in conjunction with the power of the Deep Convolutional NN really enable a **real-time classification**. The next paragraphs contain the solutions details more in depth.

3.3.5 Deep Recurrent Neural Network

The Recurrent Neural Networks work with time-dependent data. The Deep Recurrent Neural Network architecture designed and implemented to solve the TMR problem is depicted in figure 3.16. It includes an input layer referred to the layer in charge of receiving the feature vectors, in details the set of vectors of 1×74 as input; it includes 3 hidden layers of 512 LSTM cell, 128 LSTM cell and 32 LSTM respectively, and an output layer for the 7 class classification.
The configuration depicted in figure 3.16 is the result of the several different attempts looking for the best configuration able to achieve the best classification performances for the complete set of transportation modalities.
The trained Deep Neural Network in question returns an algorithm model (also called executable model) consisting in a set of values created by the Deep Neural Network itself as representation of the correlation between input and output data. This algorithm model will be able to classify further data starting from the extracted features of the same form of what is given in input during training process. Basically, the algorithm model will expect vectors in the form of 1×74, then corresponding to a time windows of data collection of about 2 seconds, for providing a eligible prediction.

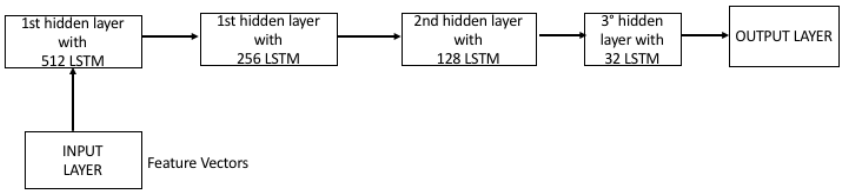

Fig. 3.16 Deep Recurrent Neural Network Architecture

3.3.6 Deep Convolution Neural Network

The Convolutional Neural Networks are able to create autonomously the feature maps as result of a correlation between the input data. To do that they entail the presence of a set of filters used for convolution process.
The Convolutional Neural Network, as already discussed in the previous chapters are particularly suited for image recognition. This means that the input size should be in form of an image and then $m \times n$ (with m rows size and n columns size). For this reason the input dimension of each vector in this context will have the form of 64×4 (and then a time windows of data collection of 1.28 seconds) creating from the beginning a time-variant correlation raw data of the same modality for the same trip directly received from smartphones' sensors.
Figure 3.17 shows the Deep Convolutional Neural Network Architecture properly designed and implemented for the TMR solution. It is composed by an *Input Layer* that receives in input vectors having the form of 64×4 components, a set of 5 hidden *convolutional layers* having 14,6,8,6,8 filters (how many feature maps are obtained after the convolution for each convolutional layer) respectively. Each Convolutional layer has its own filter size and activation function as well as it is linked to a fully-connected feed-forward layer (Dense Layer) which is in charge of receiving in input the feature maps created by each convolutional layer.
The Dense Layers output is captured by a Concatenation Filter able to concatenate the acquired knowledge of each Dense Layer. At the end an *Output Layer* is set up for the 7 class classification.
The purpose of this kind of architecture is then capturing the informations at various levels and correlate them directly.
For sake of clarity, please note that also in this case the Deep Convolutional Neural Network depicted in figure 3.17 is the result of several different attempts aimed at looking for the best configuration returning the best performances for all the classes.
After the training process with the raw data the algorithm model (executable model) is then stored for further prediction.

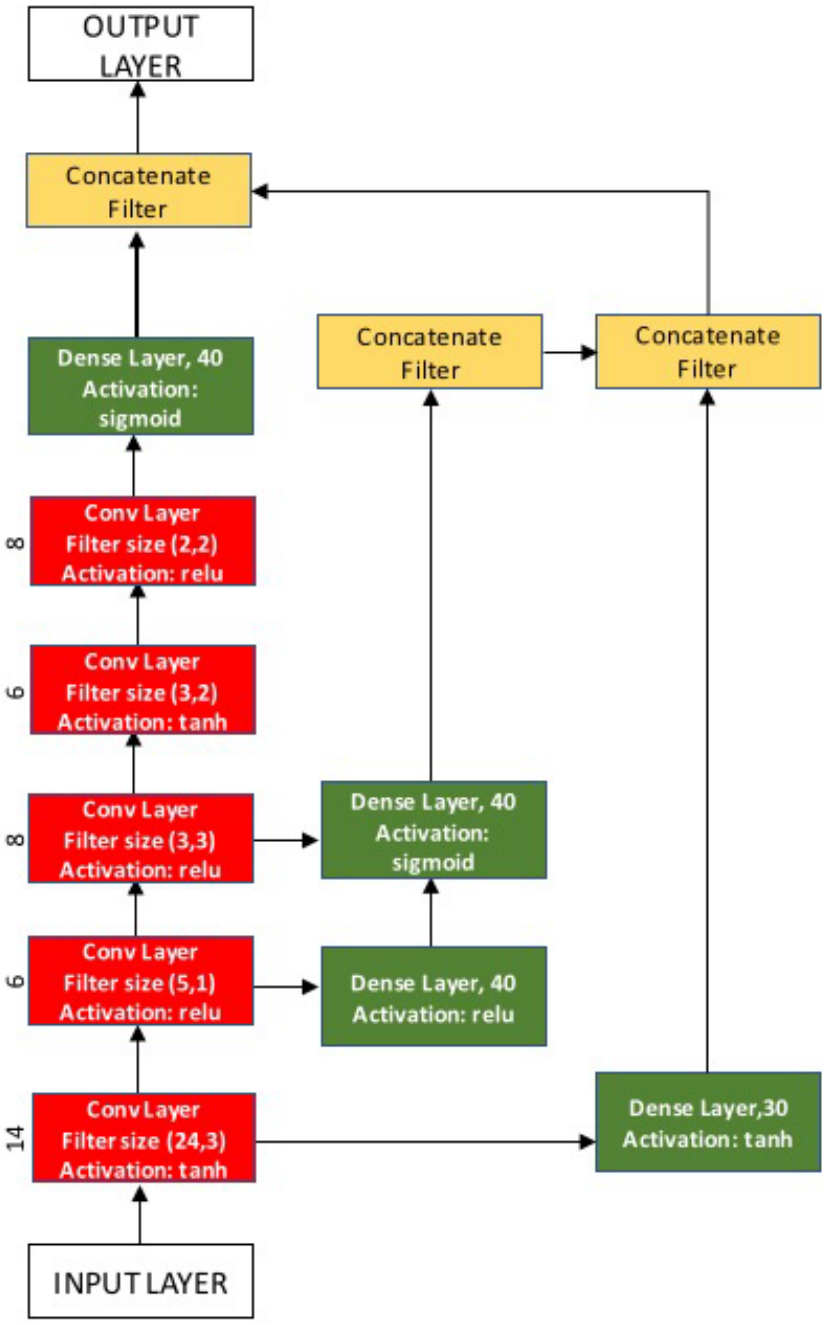

Fig. 3.17 Deep Convolutional Neural Network Architecture

3.3.7 Real Time Recognition

The Deep Learning algorithms presented above have been implemented exploiting a *Python library* [1] called *Keras* [2] and the algorithm models that hereinafter I call *Recurrent model* and *Convolutional model* have been saved by exploiting a Keras function properly implemented for this scope. The algorithm models, namely the two obtained after the training process of both Deep Recurrent and Deep Convolutional Neural Networks have been published to a remote server where both expose HTTP RestFULL APIs [3]. An Android based APP (referred to the Real Time Recognition APP as depicted in figure 3.10) was developed to (1) read sensors data information, (2) acquire the raw data, (3) send the raw data to the remote server. By exploiting the implemented APIs the two algorithm models are triggered to be able to return the expected classification taking into account the data received.

3.3.8 Results

The Deep Learning algorithms was trained with the entire dataset acquired for all transportation modalities i.e. *Car, Walk, Motorbike, Tram, Subway Bus, and Still*. A portion of 25% of the dataset was used to validate the Deep Learning algorithm during the training process. So that the algorithm could be monitoring during the learning process with the aim of find the best hyperparameter [4].
To figure out the effectiveness of the implemented solutions to solve the TMR problem two types of validation tests was performed; the first is referred to the **validation test** obtained with the validation data used during the training process. Those data have not been seen by the algorithms and then can be used for testing purposes obtaining the confusion matrices for both Deep Algorithms. The second is referred to a test directly conduced in the **real-environment**s where through the Real Time Recognition APP and involving 10 persons the results were then collected.

Validation Test

Table 3.1 and Table 3.2 show the confusion matrices (error matrices showing the algorithms performances) of the two Deep Learning algorithms. The algorithms performances are evaluate by considering Precision,Recall and F1-Score that are

[1] www.python.org
[2] www.keras.io
[3] https://restfulapi.net/http-methods/

typical indices to determine the algorithm performances in statistical learning. The precision is defined as:

$$P = TruePositive/TruePositive + FalsePositive \tag{3.26}$$

The Recall is defined as:

$$R = TruePositive/TruePositive + FalseNegative \tag{3.27}$$

The F1-Score is defined as the harmonic means of the Precision and Recall. The *True Positive* can be defined as the value where the row and the column have the same label (e.g. in Table 3.1 a 36 is a true positive for Motorbike). The *False Negative* are those values that take place in the column except for the value that has the same label in the row and column (e.g. in Table 3.1 for Motorbike the values 3 Still and 3 Car are the False Negative). The *False Positive* are those values in the row except for the value that has the same label for row and column (e.g. in Table 3.1 for Motorbike the values 3 Bus and 2 Car are the False Positive).

Table 3.1 Confusion Matrix for Deep Recurrent Neural Network

	Actual									
Predicted	Motorbike	Walk	Bus	Subway	Tram	Still	Car	Precision	Recall	F1-Score
Motorbike	36	0	3	0	0	0	2	0,878049	0,857143	0,8674699
Walk	0	41	0	0	0	3	0	0,931818	0,97619	0,9534884
Bus	0	0	38	0	0	0	0	1	0,9047	0,95
Subway	0	0	0	40	3	0	0	0,930233	0,952381	0,9411765
Tram	0	0	1	2	39	0	0	0,928571	0,928571	0,928571
Still	3	1	0	0	0	39	0	0,906977	0,928571	0,9176471
Car	3	0	0	0	0	0	40	0,930233	0,952381	0,9411765

The algorithm performances are both high as shown in Tables 3.1-3.4. The Deep Convolutional Neural Network with the validation data test seems to be better than the Deep Recurrent Neural Network of about 3% only considering the different between the avg/total for F1-Score and a percentage of recognition, according to the avg/total of F1-Score of **96,25%** for Deep Convolutional versus the **92,85%** for the Deep Recurrent. The accuracy in classification of the new data is higher than the other results obtained with other algorithms and methodologies already presented in the literature.

Table 3.2 Confusion Matrix for Deep Convolutional Neural Network

	Actual									
Predicted	Motorbike	Walk	Bus	Subway	Tram	Still	Car	Precision	Recall	F1-Score
Motorbike	38	0	2	0	0	0	2	0,9047	0,9047	0,9047
Walk	0	42	0	0	0	3	0	0,93	1	0,9655
Bus	0	0	40	0	0	0	0	1	0,95	0,97
Subway	0	0	0	42	0	0	0	1	1	1
Tram	0	0	0	0	42	0	0	1	1	1
Still	2	0	0	0	0	39	0	0,9512	0,9285	0,9397
Car	2	0	0	0	0	0	40	0,9523	0,9523	0,9523

Table 3.3 Average Total Performances for Deep Recurrent Neural Network

	Precision	Recall	F1-Score
avg/total	0,929411	0,928571	0,9285042

Real-Environment Test

The Real-Environement Test have been conduced with real users using the Real-time Recognition APP (RTRA) during their travel.

The RTRA is in charge of collecting during the users travel at most 2 seconds, for each data take-over, of raw data. The RTRA sends the collected data to the server by exploiting the implemented APIs for classification purposes.

The two classificators, namely the Deep Recurrent Neural Network and the Deep Convolutional Neural Network, use the needed data for providing the classification and then 2 seconds for the first classificator (Deep Recurrent) and 1.28 seconds for the second classification (Deep Convolutional). The server where both algorithms are implemented return the modes recognized.

For each transport modalities at least 100 tests was realized and the performance results for both algorithms are reported in table 3.5 for features based and table 3.6 for raw data based.

According to the performance results described by the tables 3.5 and 3.6 the best performances in terms of accuracy in recognition are performed by the Deep Convolutional Neural Networks, and also it is able to achieve such performances having a shortest time window of data collection and it doesn't need features process.

Table 3.4 Average Total Performances for Deep Convolutional Neural Network

	Precision	Recall	F1-Score
avg/total	0,963099	0,962585	0,9625756

Table 3.5 Real Environment Performances for Deep Recurrent Neural Network

Feature Based Deep Recurrent	Recognition %
Motorbike	48
Walk	99
Bus	93
Subway	92
Tram	93
Still	99
Car	92
avg/total	88

Table 3.6 Real Environment Performances for Deep Convolutional Neural Network

Feature Based Deep Recurrent	Recognition %
Motorbike	96
Walk	100
Bus	98
Subway	100
Tram	100
Still	100
Car	96
avg/total	98.6

For the sake of clarity it is important to stress that both approaches require a lot of time for the training process where the algorithm models are then created. At the end of the training process both approaches are able to return classification in a very short time with the great difference that the Deep Recurrent needs about 2 seconds of data collection and also it needs to process the raw data by extracting features in order to perform the classification; whereas the Deep Convolutional needs at most 1.28 seconds of time windows data collection for providing the classification, reaching almost real-time recognition.

3.4 Traffic Control based Future Works

A potential future work that I have preliminary investigated, is able to exploit the implemented TMR solution to bring awareness Traffic Control.
In fact, as already introduced the TMR solution could be adopted by several transportation management providers and in particular by the traffic management departure.
The TMR solution, presented in the previous paragraphs, permits of identifying the transportation means in urban and extra urban area, while people are moving on the streets. For instance, figure 3.18 shows the motion flows of two transportation modalities (*Car* in blue and *Motorbike* in orange, and the axis represent the Latitude and Longitude) identified with the Real Time Recognition APP for testing purposes. Starting from the real time recognition

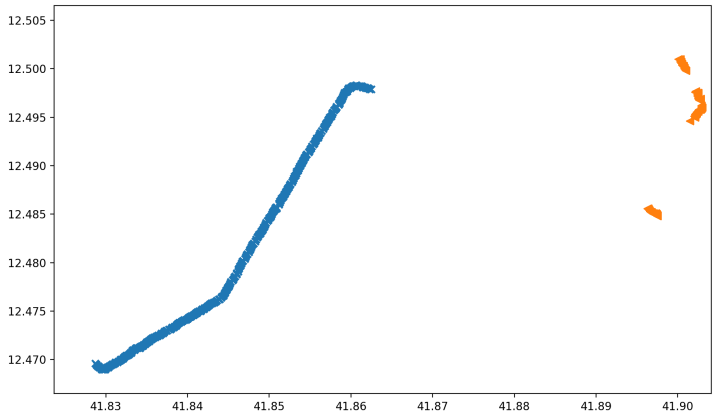

Fig. 3.18 Car and Motorbike Motion Flow in Rome

by exploiting the implemented Deep Convolutional Neural Network solution

is possible to identify the people motion flows. The motion flow determines how a recognized transport modality is moving on the streets and with which velocity. In this respect the Real Time recognition APP is able to monitor the speed, Latitude, Longitude and recognize the transport modality.
Recent studies [65] [66] [106] try to improve the efficiency of the traffic lights providing multiple solutions for managing the traffic lights signals. The three mentioned solutions, [65] [66] and [106], to perform the traffic lights control use a Deep Reinforcement Learning [103] approach for controlling the traffic light signals, by using sensors on the streets, camera sensors positioned in proximity of the traffic lights, vehicular networks [42], and traffic data collection.
The most promising and recent approach (at least from my perspective) is described in [66] where the authors propose a Deep Reinforcement Learning solution with a Deep Convolutional Neural Network able to process the real time input data and the Reinforcement Learning algorithm able to decide the timing for the traffic light signals. This approach entails the presence of sensors equipped in each traffic light to monitor the status of the vehicular network as for instance number of vehicles, the location of vehicles, and their waiting time. By using the TMR solution basic input data $<position,speed>$ to perform the traffic lights control as described in [66] can be used with the transportation modality by optimizing the traffic light signals also according to the type of transportation means on the streets. In fact, the basic input data as number of vehicles, the location of vehicles, and their waiting time can be estimate by processing the recognition from real-time TMR solution.

Chapter 4

Deep Model Predictive Control based techniques

The problem of controlling the glucose level for diabetic patients is the starting point for most of the available treatments. Despite the importance of this problem, the complex dynamics that regulate the evolutions of the glycaemic index and many of connected biological factors, force the solutions commonly implemented in medical devices as for instance artificial pancreas to make several simplifying assumptions. Several different approaches have widely investigated this problem, implementing control solutions spacing from traditional threshold based ones to PID and MPC.
In this chapter a new solution is presented for improving the State-of-the-art algorithms performances and methodologies proposing a work inspired to what developed in [112], [80], and [74]. The proposed solution is based on Bidirectional Neural Network and Model Predictive Control techniques aimed at maintaining the blood glucose level within a safe range.
The Deep Bidirectional Neural Network are mainly used for time series prediction evaluating the evolution of the blood glucose signal, and the Model Predictive Control scheme is used as optimization strategy for controlling the biological factor of interest (blood glucose level) via insulin injection quantity.
The approach reported in this chapter aims to highlight the results obtained with the Deep Neural Network techniques since the Model Predictive Control controller solution is still under a finishing process. Therefore, the MPC solution is presented as objective that will be reached when the work will be indeed completed.

4.1 Blood Glucose Level Control

The design and control of a system dedicated to the blood glucose level control is widely investigated in literature. In [79] the authors have proposed a solution with a model based predictive control algorithm for controlling the normoglycemia in diabetic patients with diabetes of Type 1 by using a closed loop insulin injection; they also develop a compartmental modeling.
The authors in [20] develop a system of a closed-loop control for glyceamic index in diabetes patients by combining the closed-loop control algorithm a glucose sensor and an actuator for insulin infusion. In [92] the authors propose a closed-loop insulin delivery system with an associated subcutaneous glucose sensor and related subcutaneous insulin injection; this solution for controlling the glycaemic index was experimented over 10 patients afflicted by diabetes of type 1.
In [33] the authors have introduced a solution based on a Bergamn nonlinear mathematical model Bergman describing the the plasma insulin injection; a semi closed loop algorithm is presented for the correction of hyperglycemia in diabetic patients. The performances of such an approaches was then evaluated with a simulation for testing the effectiveness.
The glucose control have been also investigated in several other different works, following the methodologies presented achieving important results as described in [60], [96], [83], [6] and [27]. In these works the authors have presented solutions dealing with closed loop control algorithms aimed at achieving desired level of blood glucose level with respect to the actual level; the main problem in closed loop based algorithms is that this methodology adapts its control actions following the biological signals without any prediction of further effects and it is has not optimization properties.
The Model Predictive Control methodologies imposes an iterative open loop optimization following the residing horizon paradigm, hence bringing the advance of the closed loop control to the optimality of the open loop. The recent and most advanced solution in this field, at least from my knowledge, is presented in [74]; in this work the authors have presented a model predictive control based algorithm for implementing an artificial pancreas for a personalized treatment of type 1 diabetes, providing several personalized control schemes for blood-glucose concentration-regulation. The work in [74] is based on synthetic data, generated with the UVA/PADOVA simulator [71], a tool recognize bu

the U.S Food and Drug Administration as a substitute to animal trials in preclinical testing.

4.2 Model Predictive Control

Model Predictive Control (MPC) [15] is an advanced control methodology used in several application contexts where there is a need to respect some constraints. It deals with large-scale systems and with a relevant number of control variables. The MPC is in charge of proving a method treating with constraints on input and states. The optimization problem and the control strategies are defined in discrete time domain; typically for an efficient implementation of the MPC solution a model of the process and cost function that must be optimized are required.

Now, let us to consider a discrete-time linear system:

$$x(k+1) = Ax(k) + Bu(k) + Md(k) \tag{4.1}$$

the $x(k) \in R^n$ represent the state vector, the input vector in 4.1 is represented by $u(k) \in R^m$, and the disturbance vector in equation 4.1 is represented with $d(k) \in R^l$, at the kth sampling time instant. According to equation 4.1 the system matrices are $A \in R^{n\times n}$, $B \in R^{n\times m}$ and $M \in R^{x\times l}$. With N we can identify the prediction horizon.
Let us to consider a predicted input sequence with the form:

$$U(k) = [u^T(k|k), u^T(k+1|k), ..., u^T(k+N-1|k)]^T \tag{4.2}$$

and we have also consider the disturbance sequence having the form:

$$D(k) = [d^T(k), d^T(k+1), ..., d^T(k+N-1)]^T \tag{4.3}$$

It is possible to obtain the time evolution of the state by simulating the equation in 4.1 forward for N sampling time intervals having the initial condition $x(k|k) = x(k)$ [74].Accordingly,

$$X(k+1) = [x^T(k+1|k), x^T(k+2|k), ..., x^T(k+N|k)]^T \tag{4.4}$$

the input and state at time $k+i$ with $i \in N$ predicted at time k have the form

$$u(k+i|k) \tag{4.5}$$

$$x(k+i|k) \tag{4.6}$$

The system that must be controlled receives a control law generated by solving the optimization problem following a defined and dedicated cost function that must be minimized, as for instance:

$$J(x(k), U(\cdot), k) = \sum_{i=0}^{N-1} ||x(k+i|k) - x_{ref}(k+i)||_Q^2 + ||u(k+i|k) - u_{ref}(k+i)||_R^2 \tag{4.7}$$

where $x_{ref}(k)$ represents the states at time k and $u_{ref}(k)$ represents the inputs references at time k. The mentioned references vectors are included in:

$$U_{ref}(k) = [u_{ref}^T, u_{ref}^T(k+1), ..., u_{ref}^T(k+N-1)]^T \tag{4.8}$$

and

$$X_{ref}(k) = [x_{ref}^T, x_{ref}^T(k+1), ..., x_{ref}^T(k+N-1)]^T \tag{4.9}$$

The Q and R matrices in equation 4.7 are the symmetric positive definite matrices.
The optimal control sequence $U^o(k)$ has the form:

$$U^o(k) = argmin_U J(x(k), U(\cdot), k) \tag{4.10}$$

At the end, once obtained the generation of the control input [74] and taking into account the receding horizon approach, the first element of the $U^o(k)$, considered to be the optimal control sequence, is then applied to the process $u(k) = u^o(k|k)$. The obtained optimized procedure has to be repeated for each sampling time k.
Yet, it is clear that one of the most important characteristic of the Model Predictive Control is the presence of the input ans state constraints in the optimization problem. In the MPC problem formulation we have both equality constrains representing the equation in 4.1 as the model dynamics and inequality constraints on input and state variables.Typically, the equality constraints are used to predict state trajectories while the inequality constraints are imposed in the optimization problem. The equation 4.7 defined as cost function can

be endowed with the state prediction at the horizon N [74]. Hence, with the mentioned modification of equation 4.7 is possible to obtain a quadratic cost function having the form:

$$\begin{aligned} J(x(k), U(\cdot), k) = \sum_{i=0}^{N-1} ||x(k+i|k) - x_{ref} \\ (k+i)||_Q^2 + \\ +||u(k+i|k) - u_{ref}(k+i)||_R^2 + ||x(k+N|k)||_P^2 \end{aligned} \tag{4.11}$$

The equation in 4.11 is hang out to the state dynamics presented in 4.1, where the presence of P being the unique nonnegative solution of the discrete time Riccardi equation [74]

$$P(k) = Q + A^T P(k+1)A - A^T P(k+1)B \times (R + B^T P(k+1)B)^{-1} B^T P(k+1)A \tag{4.12}$$

The matrix $P \in R^{n \times n}$ is the above mentioned weight matrix related to $x(k + N|k)$; such form represent the predicted state over the horizon N. Hence considering the horizon N, the predicted state trajectories of the system dynamics take the form:

$$X(k+1) = \mathcal{A}x(k) + \mathcal{B}U(k) + \mathcal{M}D(k) \tag{4.13}$$

The matrices in the above equation, as described in [74], $\mathcal{A} \in R^{nN \times n}$, $\mathcal{B} \in R^{nN \times mN}$, $\mathcal{M} \in R^{nN \times lN}$ are then obtained by using algebraic calculation staring from the equation in 4.1. Depending on the problem or kind of application $D(k)$ can be either estimated or known/unknown. Whether the disturb is unknown, the MPC calibration achieved with the symmetric definite positive matrices Q and R, must be robust enough to guarantee at least as suboptimal control performances. The equation in 4.11 considered to be the quadratic cost function under these assumption takes the form:

$$\begin{aligned} J(x(k), U(\cdot), k) = ||U(k)||_H^2 + 2(x^T(k)\mathcal{F}_x^T + D^T(k)\mathcal{F}_D^T - \\ -U_{ref}^T(k)\mathcal{R} - X_{ref}^T(k)\mathcal{F}_{ref}^T)U(k) \end{aligned} \tag{4.14}$$

where only the terms depending on $U(k)$ have been maintained and

$$\mathcal{H} = \mathcal{B}^T \mathcal{Q} \mathcal{B} + \mathcal{R} \tag{4.15}$$

$$\mathcal{F}_x = \mathcal{B}^T \mathcal{Q} \mathcal{A} \tag{4.16}$$

$$\mathcal{F}_D = \mathcal{B}^T \mathcal{Q} \mathcal{M} \tag{4.17}$$

$$\mathcal{F}_{X_{ref}} = \mathcal{B}^T \mathcal{Q} \tag{4.18}$$

where $\mathcal{Q} = diag(Q, ..., Q) \in R^{nN \times nN}$ and $\mathcal{R} = diag(R, ..., R) \in R^{mN \times mN}$.
Whether the input and state constraints are not considered in the optimization problem, under the assumption of nonsingularity of the matrix $\mathcal{H}$, the solution exist is unique and has the form:

$$U^o(k) = \mathcal{H}^{-1}(-\mathcal{F}_x x(k) - \mathcal{F}_D D(k) + \mathcal{R} U_{ref}(k) + \mathcal{F}_{x_{ref}} X_{ref}(k)) \tag{4.19}$$

On the contrary, whether the input and state constraints are considered, the optimization problem can be solved via quadratic programming optimizer [74].

4.3 Deep Model Predictive Control

Model Predictive Control is widely used for addressing various control complex tasks in different fields such as mobile robot maneuvering, humanoid robots. As already discussed in section 4.2, MPC has a fundamental advantage with respect to closed loop control techniques that since it has an accurate predictive model permits to optimize the control inputs with properly designed defined cost function. In MPC a key challenge is to identify, with a reasonable accuracy, the dynamic model especially in presence of highly non-linear systems.
The authors in [64] have developed a system for controlling a PR2 robot able to cut food items. In this context, the dynamic of the system depends by several different properties such as the type of food that should be cut, the coefficient of friction with the knife, fracture effects and elastic modulus. In this work, presented in [64] a key challenge is to design a model that is able to take into account all the different properties discussed above; for this reason the authors defined the design of the model as unfeasible.
To solve the mentioned issues and for controlling a food cutting PR2 robot, they have implemented a system that combines deep learning system with an MPC controller. The use of deep learning in essential in their work since deep learning techniques are able to learn non-linear, time-varying dynamics while remaining differentiable using back-propagation algorithms.
Basically, the authors in [64] divide the approach in two steps; in the first step

they implement a deep learning architecture able to provide model parameters by acquiring knowledge from a dataset which contains a huge set of examples for robotic cutting towards 20 different material types. The output of the first step are a set of parameters able to estimates the model dynamics. In the second step they use the learned dynamics for predicting the future state via recurrent prediction and use it for implementing a real-time model predictive control able to accomplish task-specific control.

Figure 4.1 shown the functional architecture of the DeepMPC system proposed

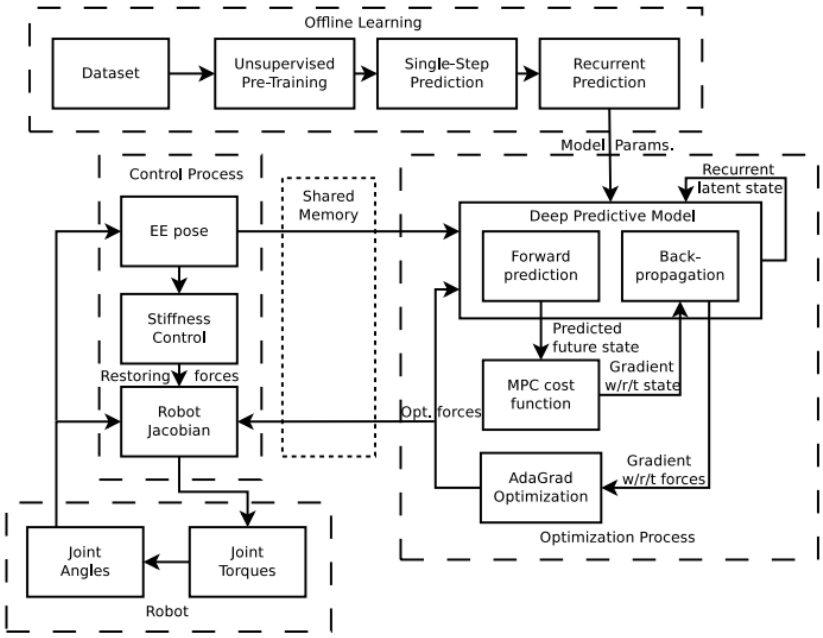

Fig. 4.1 DeepMPC system [64]

in [64]. The system is divided into 4 blocks concerning the robot with joint angles and joint torques, the control process, the offline learning that performs the learning procedure from dataset up to the model parameters, and optimization process with the real time model predictive control.

Basically, the control process, provide to the optimization process the robot's joint angles pose, the pose is used by the forward prediction, which exploiting the learned model parameters as presented in equation 4.20, to provide the *Predicted future state* for computing the gradients with respect to the state of the MPC cost function back-propagating it to the deep predictive model and obtaining the gradients with respect to the forces. They use an AdaGram [29] optimization to compute the optimized forces providing them to the control

process. The MPC cost function for such task, that is food cutting is presented in equation 4.21 as describe in [64].

$$\Theta = (W^{[f]}, W^{[c]}, W^{[l]}, W^{[o]}, W^{[lp]}, W^{[lc]}, W^{[ll]}, W^{[lo]}) \tag{4.20}$$

$$C(X, U) = C_{cut}(X) + \beta C_{saw}(X) \tag{4.21}$$

The first term in equation 4.21, that is expressed in 4.22, represents the driving for the controller to move the knife and

$$C_{cut} = \sum_{k=t}^{t+T} P_z^k + \gamma P_z^{t+T} \tag{4.22}$$

The second term in 4.21, that is expressed in 4.23, is able to maintaining the tip of the knife within a "sawing range" [64].

$$C_{saw} = \sum_{k=t}^{t+T} (max\{0, |P_x^k - P_x^*| - d_s + \lambda\})^2 \tag{4.23}$$

In equation 4.23 the P_x^* is the center point for the sawing range, and the d_s is the range as well as λ represent the margin.

The presented work developed by the authors in [64] is an example of how is possible to combine the Deep Learning and MPC into a unique solution. This will lay the basis for our proposed solution for controlling the biological factors in diabetic patients as will be presented in the next paragraphs.

4.4 DeepMPC Artificial Pancreas

In the context of managing metabolic disorder a relevant introduction about the problem is expressed in [74]:

"*T1D is a metabolic disorder that causes a total insulin deficiency and impedes the regulation of blood glucose concentration (also called glycemia) by the pancreas. Insulin is an essential hormone for blood glucose control, and the lack of insulin could result in chronic hyperglycemia, which exposes T1D subjects to risky long-term complications. The lack of insulin must be compensated with exogenous, usually subcutaneous, insulin administration that can result in hypoglycemia if the amount of insulin is overestimated. Hypoglycemia is associated with short-term complications that, in severe cases, can result in a*

coma or death. Consequently, T1D subjects face the challenge of maintaining their glycemia within a safe range."

The development of an *Artificial Pancreas* in charge of managing, in an intelligent manner, the regulation of insulin released is a basic concept to make progress in the diabetic disorder. Nowadays, some people still use the direct insulin release by using, manually, the syringe in time and days as fixed by physicians.
The Artificial Pancreas creates the possibility to discard the manually distribution of insulin with fixed dosages and enables the possibility to use an automatic release of insulin when needed.
The approach discussed in this section deals with the possibility to create an Artificial Pancreas system able for automatic subcutaneous insulin pumps injection. Currently, several different closed loop algorithms have been implemented and tested towards portable devices. But this approach (the closed loop one) is affected by inherent delays [74] and then it requires other and more efficiency techniques. Model Predictive Control can be used as control technique in this context and it is already used in several different clinical trials.
In the problem of controlling biological factors such as blood glucose control the subject dynamics has the most important role. In fact, in diabetic patients we can find different glucose-insulin dynamics, and the control is build over the artificial pancreas has to be robust, reliable and safe to guarantee acceptable performances.
Why different dynamics? Because each subject has its own physiological characteristic that could produce different dynamics. In this respect, following the Model Predictive Control principle that produces control actions according to a cost function model-based, for each subject a proper model must be included in order to produce acceptable control actions and hence acceptable performances from Artificial Pancreas.
The design of a individualized model for each subject - i.e. diabetic patient - is a hard task; even because the diabetic model patients is highly non-linear and

time variant (see equation 4.24 as reported in [74]).

$$\begin{cases} \dot{x}_1(t) = -k_{gri}x_1(t) + d(t) \\ \dot{x}_2(t) = k_{gri} - k_{empt}(x_1(t) + x_2(t))x_2(t) \\ \dot{x}_3(t) = -kabsx_3(t) + k_{empt}(x_1(t) + x_2(t))x_2(t) \\ \dot{x}_4(t) = EGP(t) + Ra(t) - U_{ii}(t) - E(t) - k_1x_4(t) + k_2x_5(t) \\ \dot{x}_5(t) = -U_{id}(t) + k_1x_4(t) - k_2x_5(t) \\ \dot{x}_6(t) = -(m_2 + m_4)x_6(t) + m_1x_{10}(t) + k_{a1}x_{11}(t) + k_{a2}x_{12}(t) \\ \dot{x}_7(t) = -p_{2U}X_7(t) + P_{2U}(X_6(t)/V_I - I_B) \\ \dot{x}_8(t) = -k_ix_8(t) + k_ix_6(t)/V_I \\ \dot{x}_9(t) = -k_ix_9(t) + k_ix_8(t) \\ \dot{x}_{10}(t) = -(m_1 + m_3(t))x_{10}(t) + m_2x_6(t) \\ \dot{x}_{11}(t) = -(k_d + k_{a1}x_{11}(t) + i(t) \\ \dot{x}_{12}(t) = k_dx_{11}(t) - k_{a2}x_{12}(t) \\ \dot{x}_{13}(t) = -k_{sc}x_{13}(t) + k_{sc}x_4(t) \\ \dot{x}_{14}(t) = -n_Gx_{14}(t) + SR_H(t) \\ \dot{x}_{15}(t) = -k_Hx_{15}(t) + k_Hmax\{x_{14}(t) - H_b, 0\} \\ \dot{x}_{16}(t) = \dot{S}R_H^S(t) \end{cases} \tag{4.24}$$

In [74] the authors have considered three individualization techniques to summarize a Model Predictive Control based linear model. In order to linearize the model in 4.24 the metabolic parameters in the model 4.24 e.g., Time-varying parameters $k_{empt}, Ra(t), E(t), U_{id}, etc.$ have been substituting with their average values. The average parameters are hence included in the model, linearizing it around a fictitious basal equilibrium [74]; after that by considering the other values (i.e., model inputs) $i(t), d(t)$ to be $i(t) = i_b(t)$ and $d = 0$; the model become the linear model having the form:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + Md(k) \\ y(k) = Cx(k) \end{cases} \tag{4.25}$$

where the control action is expressed by $u(k)$ as insulin injection and $d(k)$ is the variation of insulin infusion as subcutaneous administration. To customize the model, starting from the linearized one i.e. 4.25, can be reached by substituting

some input parameters depending on the subject as expressed in [74].

The linearized model representing a complex subject behaviour, even if tailored by some additional modification, loses a lot of information that are indeed useful for controlling purposes. The lost of that information will be reflected into the performances of such approach, and in this respect the main goal of my current research activities is to try to approximate the highly non-linearity of the complete model (see equation 4.24) by using Deep Learning methodologies, in order to reach high precision through data driven control.
By exploiting the UVA/PADOVA [71] in-silico tool that is able to represent diabetic patients and their behavior during the administration of meals (that means the glycemic index after minutes/hours from the meal), I have approximated the non-linear and time variant diabetic user model by training a Bidirectional Recurrent Neural Network and then obtaining a *model* (output from the traning process) that is able to return glycemic index level with respect to a given meal for a specific subject.
Inspired by the DeepMPC approach presented in the previous section, I have designed an artificial pancreas approach based on the Deep Learning solution where the biological factors are controlled by a Model Predictive Control.
Figure 4.2 shown the DeepMPC architecture designed for building an innova-

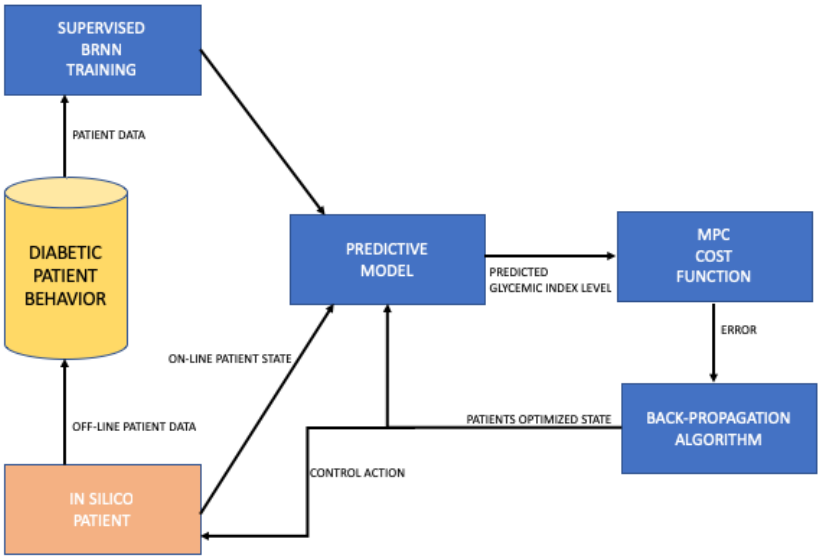

Fig. 4.2 DeepMPC architecture

tive artificial pancreas. The UVA/PADOVA tool [71] contains a set of simulated

diabetic patients; the simulation consists in providing reliable biological factors from patients after that a set of meals are dispensed. In the tool there is the possibility, also, to assign meals, and get decisions in terms of carbohydrates assumed in each meals. The UVA/PADOVA tool already includes a closed-loop control algorithm that taking into account a specific user and given meals, provides at the beginning of each meal the corresponding dose of bosul insulin to be injected; moreover, there is the possibility to see users' behavior in case of open-loop control, i.e. enabling a fixed control regardless of the type of meal (the fixed control consists of bosul insulin to be injected) or disable it altogether.
By exploiting the tool we have created a diabetic patient dataset (see its partial content in Table 4.1) that contains the patient's behaviour, i.e., Glucose Level index in the table 4.1, during the meals simulating 1 year with both closed-loop and open-loop control, and considering 5 meals per day with different quantity of carbohydrates per meal. The Table 4.2 represents an example of the given

Table 4.1 Partial content of the dataset

Time	Glucagn	Meal	DestroseIV	Insulin_IV	Insulin_sub_Injection	Glucose_Level
.....	.....	.....	.....	.....	.....	.....
359	0	0	0	0	120	134,3059474
360	0	3333,333333	0	0	11520	134,3109421
361	0	3333,333333	0	0	120	134,3161604
362	0	3333,333333	0	0	120	134,3237609
363	0	3333,333333	0	0	120	134,3353563
364	0	3333,333333	0	0	120	134,3520073
365	0	3333,333333	0	0	120	134,3768196
366	0	3333,333333	0	0	120	134,4138789
367	0	3333,333333	0	0	120	134,4678387
368	0	3333,333333	0	0	120	134,5439337
369	0	3333,333333	0	0	120	134,6485899
370	0	3333,333333	0	0	120	134,7922252
371	0	3333,333333	0	0	120	134,9982503
372	0	3333,333333	0	0	120	135,3169522
373	0	3333,333333	0	0	120	135,8177187
374	0	3333,333333	0	0	120	136,5554574
375	0	0	0	0	120	137,5588808

meals for the simulated diabetic patient where the administration of the carbohydrates is considered being variable during the days (e.g. 10-45 for breakfast means that the administration of carbohydrates could vary in the range of 10

and 45).

The DeepMPC approach introduced in this work starts with the analysis of

Table 4.2 Variable Meals for specific patients

	breakfast(7am)	lunch(1pm)	snack(4pm)	dinner(7pm)	snack(11pm)
carbs[g]	10-45	55-130	5-30	45-140	10-30

the dataset by training a supervised Biderectional Recurrent Network that shall produce in output a personalized (i.e. a model dedicated to a specific diabetic patient) trained model able to predict glycemic index levels taking as input the online patient state for the duration of the meal (see table 4.3 referred to just a minute of the complete meal; the complete meals are typically considered having a duration of 15 minutes). The prediction of future glucose levels has been already investigated in [80].

The predicted glycemic index level in output concerns the glycemic index levels

Table 4.3 On-line diabetic patient state

Glucagn	Meal	DestroseIV	Insulin_IV	Insulin_sub_Injection	Glucose_Level
0	3333,333333	0	0	120	144,3109421

after 30 minutes, 1 hours and 1 hour and 30 minutes staring from the moment generated by the on-line patients state, the mentioned timing correspond to the peak glycemics (see table 4.4).

The predicted peak glycemics are the consequence of the user behavior with

Table 4.4 Glycemic index over the predicted periods

Glucose_Level_30minutes	Glucose_Level_1hour	Glucose_Level_1h_30minutes
354,7868	469,8769	513,9808

respect to the current *"GlucoseLevel"* and *"InsulinsubInjection"* depicted in table 4.3. By changing the *"InsulinsubInjection"* of table 4.3 from **120** p/mol (that correspond to basal injection [74]) to **11520** p/mol, the predictive model will return the values shown in table 4.5. This means that the predictive model will provide 3 future indexes to analyze the user behaviour after each meal; and then providing different states it is possible to observe the user behaviour in the future.

The MPC cost function will evaluate the output from the predictive model. The MPC cost function is properly designed for evaluating the error between

Table 4.5 Glycemic index over the predicted periods

Glucose_Level_30minutes	Glucose_Level_1hour	Glucose_Level_1h_30minutes
185,4003	193,7652	201,8824

the expected Target (T_i) value (the right glycemic index level for the specific user) and the Predicted value (P_i) for $i = 1, .., 3$ is:

$$y = (T_i - P_i)^2 \tag{4.26}$$

Basically the MPC cost function will evaluate the error between the Target and Predicted value.
Now, by using a back propagation algorithm the control system will find the best control action, i.e. insulin injection, to reduce as much as possible the glycemic index level over the admitted threshold (the threshold is represented by the Target values for each subject over controlling period) by minimizing the error computed with the MPC cost function. The back propagation algorithm will provide actions such that the computed error over the equation 4.26 is closest as possible to 0. Once obtained the best insulin injection (control action) for a given patient state, it is provided to the in-silico patient.

4.5 Results

This section presents the results obtained with the implemented Artificial Pancreas starting from the evaluation of the prediction values. In order to evaluate the performances I have created a dataset by using the UVA/PADOVA tool by simulating the behaviour of an in-silico diabetic patient.
Working with time-dependent data, the Bidirectional Recurrent Neural Network (BRNNs) was implemented to create the Predicted model able to synthesize the patient behaviour. The implemented architecture chosen for our testing consists in a multi-layer Bidirectional Long-Short-Term Memory composed by 3 hidden layers with 80, 60 and 50 cells respectively; the output layer is a 3 cells layer where each cell in output correspond to the number of prediction timing; in our case the 3 cells correspond to the output for 30 minutes, 1 hour, and 1 hour and 30 minutes of prediction.
The main advantage of this architecture (Bidirectional) is that during the training process, as already presented in previous section, the network learns

the user past, present and future behaviours; whereas in case of a RNN the network would have learned only the past and present behaviour with respect to a present user state.

The training process was developed for evaluating the collected data by dedicating 80% of the collected data, randomly selected among the 1 month of the data samples, to the training set, and the 20% of the collected data for validation purposes. In this way, during the learning process, it is possible to evaluate the degree of accuracy in learning the network (i.e., the accuracy in prediction future values). The accuracy in prediction during the training process evaluated with the validation test is 93%.

In order to test the performances in predicting future values starting from a

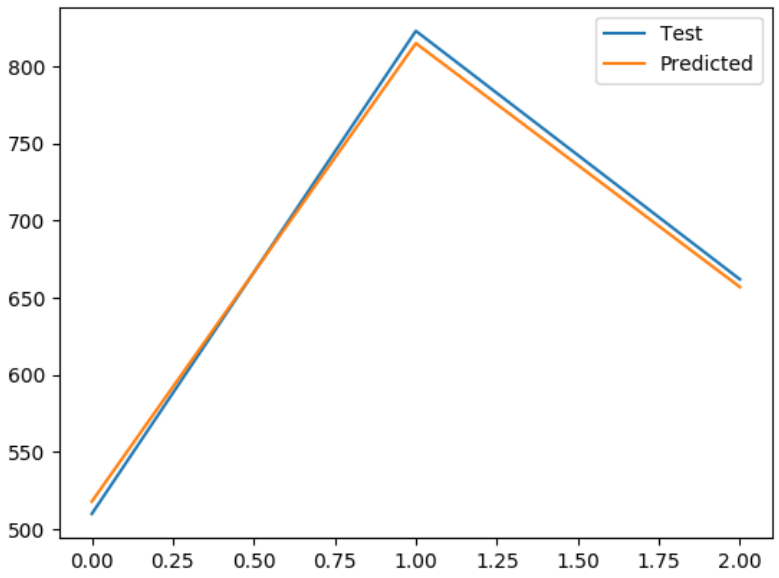

Fig. 4.3 Predicted and target values over 30 minutes, 1 hour, and 1 hour and 30 minutes

diabetic patients state, new data samples was collected by simulating 1 meal and the assumed carbohydrates for the meal is presented in table 4.6.

Figure 4.3 shown the predicted values (in orange) at 30 (0.00 on the x-axis), 1

Table 4.6 1 Meal for testing prediction

	Lunch
carb[g]	89

hour (1.00 on the x-axis) and 1 hour and 30 minutes (2.00 on the x-axis) with

the after the meal considering $89grams$ of carbohydrates and the real values (in blues) as computed with the UVA/PADOVA tool for the simulated subject. These values correspond to the measured glycemic index for the subject without any insulin injection (bolus) but just considering the basal injection [1]

The predicted values are really close to the real ones; this means that the BRNN has approximated the model subject and it is able to predicted future glycemic indices during the meal administration. Therefore, with in high level of accuracy it is possible to enable the a data-driven control.

4.6 Future Work

As already mentioned, this work is in draft version since the control part, i.e. Model Predictive Control strategy, although in an advanced state is still immature to be described and formalized as a whole. However, the way in which control will act, once the part of the Deep Learning in terms of predictive model, will provide the user's behavior, deserves to be explained. Although immature some results of the control performed with a Model Predictive Control technique can be highlighted.
In adults the normal glycemic peak should be less that 180 mg/ml considered as renal glucose threshold. In our Artificial Pancreas the quantity of insulin injection is decided dynamically taking into account the predictions in output from the BRNN. A Model Predictive Control cost function is used to determine the error (see equation 4.26) between the Target value considered as renal glucose threshold and the predicted ones.

Figure 4.4 shown the controlled in-silico patient with the implemented Artificial Pancreas for the mentioned meal. Figure 4.5 shown the controlled in-silico patient with a closed loop control implemented by the authors in [74]. The Artificial Pancreas created in this thesis provide a control able to provide an intelligent insulin injection such that the peak glycemin doesn't reach values over $200mg/dl$, while with the closed-loop control without prediction as shown in figure 4.5 the peak glycemic index is over $200mg/dl$. The glycemic values over the 5 hours in the figure 4.1 grow since the Deep MPC controller is disabled during the simulation from the 5th hour while in figure 4.5 the values are in the correct range since the closed loop controller is still active.

[1] https://www.diabetes.co.uk/insulin/basal-bolus.html

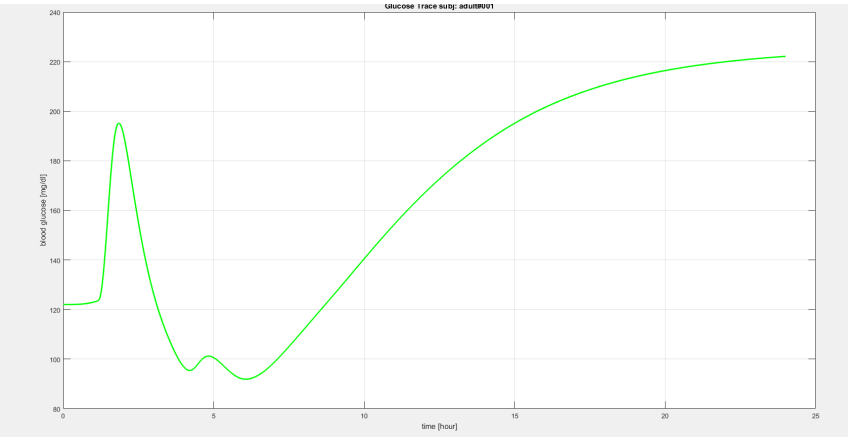

Fig. 4.4 Deep MPC Control for the administered meal

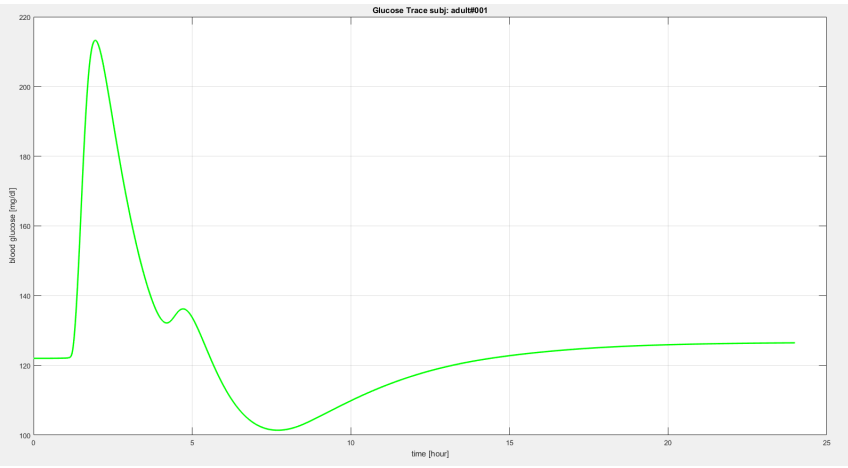

Fig. 4.5 Closed Loop Control for the administered meal

Chapter 5

Conclusion

In this PhD thesis I have reported and discussed the main research activities performed during my PhD carreer. As discussed in previous chapters, the main scope of my research activities was to bring artificial intelligence methodologies in the field of control systems employing solutions from the fields of data-driven control and intelligent control systems.
The choice of a PhD in the field of AI applied to Control System was driven by the vision of having machines perform both intelligent and reasoning actions, previously exclusive to human decisions, that are able to control complex systems for which a mathematical model may not be available.
I this context, this work reports such activities as tentative approach to use intelligent control [1] [77], *a data driven control framework*, introducing and discussing typical AI related methodologies and solutions such as Neural Network, Machine Learning in general and Reinforcement Learning.
Data-driven control researches found in the literature are often based on the process, referred to as data driven modeling, of analysis the historical data for a generic system. The goal of such modelling approach is to find the relations between the system variables (input, internal state and output) without having a priori knowledge of the physical dynamics of the system.
Machine Learning techniques are able of extracting patterns from high dimensional data discovering models of dynamical systems or learning control laws, directly from the data and eventually their experience.
In this work I have discussed that some traditional control methods could not address some systems related to real-world scenarios, due to the intrinsic complexity of the systems, as, for example, self-driving cars.
In this context, the modern solutions brought from model-free approaches such

as Reinforcement Learning, Deep Learning and Machine Learning in general (i.e. modern AI solutions), can address the possibility to control systems in which we might have unknown, or hard to model, dynamics, or system characterized by heavily nonlinear laws with high dimensional states and limited measurements. The control based machine learning can be designed thanks to the ability of the employed AI methods to identify the dominant patterns and relations that emerge from the data.

The dominant patterns identified with such machine learning methodologies directly from the data represent in a certain sense what we should control, creating hence a data-driven model that is a representation of the system itself.

In this PhD thesis, the research activities that I have conducted, have been described in order to addressing some real-world scenarios such as telecommunication networks, transportation systems and health systems for Personalized Medicine arriving to, at least from my perspective, meaningful conclusions.

In chapter 2 I have presented the approaches related to *Reinforcement Learning*, *Game Theory* and *Multi-Agent Reinforcement Learning*. More precisely, the methodologies introduced in the first part of the Chapter 2 have been investigated and properly extended in order to be applied in telecommunication networks. In the telecommunication networks such methodologies were presented to cope with multiple users involved in sharing bandwidth and an heuristic *Multi Agent Reinforcement Learning* approach was designed and implemented to satisfy the users' requests. In the scenario considered, the telecommunication network has to be able to satisfy all the users expectation, assuming bandwidth with limited capacities, and the objective is to assure suitable performances. Section 2.3 shows the results of the publication [22].

In Chapter 2 I have also presented the research activities conducted in the framework of BONVOYAGE [1] H2020 project for addressing the possibility to introduce artificial intelligence techniques in the field of Smart Transportation Systems [25] - [54] - [100]. More precisely, Section 2.4 reports: (i) a publication [17] in the field of Machine Learning methodologies to identify travellers' profiles for computing tailored trips and (ii) a *Reinforcement Learning* algorithm properly designed and implemented to rank tailored journey solutions according to a new deducted travellers' model.

In Chapter 3 I have introduced Deep Learning techniques, architecture and algorithms. A new methodology to solve the transportation mode recognition

[1] BONVOYAGE H2020 http://www.bonvoyage2020.eu/

problem is presented by describing the solution and the algorithms involved to solve the problem.
The transportation mode recognition will be used for monitoring the traffic flows enabling the possibility to control traffic lights in Vehicular Networks as presented at the end of this chapter; the traffic lights control has been presented as a future work. The performance of the presented solution is discussed in details in terms of accuracy in the classification/recognition problem.
In Chapter 4 I have presented the research activities carried out in the field of Personalized Medicine. A first study on "Deep Model predictive control",i.e. a particular approach designed for combining Deep Learning and Model Predictive Control, is presented aiming at controlling the biological factors of diabetic patients.

References

[1] Antsaklis, P., Passino, K., and Walker, I. (1994). An introduction to intelligent and autonomous control. In *IEEE Proceedings*, volume 82, pages 955–955. [New York, NY]: Institute of Electrical and Electronics Engineers,[1963-.

[2] Arel, I., Rose, D. C., and Karnowski, T. P. (2010). Deep machine learning - a new frontier in artificial intelligence research [research frontier]. *IEEE Computational Intelligence Magazine*, 5(4):13–18.

[3] Arentze, T. A. (2013). Adaptive personalized travel information systems: A bayesian method to learn users' personal preferences in multimodal transport networks. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1957–1966.

[4] Bardenet, R., Brendel, M., Kégl, B., and Sebag, M. (2013). Collaborative hyperparameter tuning. In *International Conference on Machine Learning*, pages 199–207.

[5] Battilotti, S., Canale, S., Priscoli, F. D., Fogliati, L., Lisi, F., Gori, C. G., Monaco, S., and Suraci, V. (2015). A dynamic approach to quality of experience control in cognitive future internet networks. In *European Conference on Networks and Telecommunications (EU-CNC), Paris (FRANCE)*.

[6] Bequette, B. W. (2005). A critical assessment of algorithms and challenges in the development of a closed-loop artificial pancreas. *Diabetes technology & therapeutics*, 7(1):28–47.

[7] Bhasker, J. and Samad", T. ("1991"). "the clique-partitioning problem". *"Computers e Mathematics with Applications"*, "22"("6"):"1 – 11".

[8] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[9] Böhning, D. (1992). Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200.

[10] Bouhana, A., Zidi, A., Fekih, A., Chabchoub, H., and Abed, M. (2015). An ontology-based cbr approach for personalized itinerary search systems for sustainable urban freight transport. *Expert Systems with Applications*, 42(7):3724 – 3741.

[11] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[12] Brooks, P. and Hestnes, B. (2010). User measures of quality of experience: why being objective and quantitative is important. *IEEE network*, 24(2).

[13] Brunnström, K., Beker, S. A., De Moor, K., Dooms, A., Egger, S., Garcia, M.-N., Hossfeld, T., Jumisko-Pyykkö, S., Keimel, C., Larabi, M.-C., et al. (2013). Qualinet white paper on definitions of quality of experience.

[14] Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Trans. Sys. Man Cyber Part C*, 38(2):156–172.

[15] Camacho, E. F. and Alba, C. B. (2013). *Model predictive control*. Springer Science & Business Media.

[16] Canale, S., Cimorelli, F., Facchinei, F., Gambuti, R., Palagi, L., and Suraci, V. (2015). Profiled qoe based network controller. In *2015 23rd Mediterranean Conference on Control and Automation (MED)*, pages 1085–1091.

[17] Canale, S., Giorgio, A. D., Lisi, F., Panfili, M., Celsi, L. R., Suraci, V., and Priscoli, F. D. (2016). A future internet oriented user centric extended intelligent transportation system. In *2016 24th Mediterranean Conference on Control and Automation (MED)*, pages 1133–1139.

[18] Castrucci, M., Cecchi, M., Priscoli, F. D., Fogliati, L., Garino, P., and Suraci, V. (2011). Key concepts for the future internet architecture. In *Future Network & Mobile Summit (FutureNetw), 2011*, pages 1–10. IEEE.

[19] Celsi, L. R., Battilotti, S., Cimorelli, F., Giorgi, C. G., Monaco, S., Panfili, M., Suraci, V., and Priscoli, F. D. (2015). A q-learning based approach to quality of experience control in cognitive future internet networks. In *2015 23rd Mediterranean Conference on Control and Automation (MED)*, pages 1045–1052.

[20] Cobelli, C., Renard, E., and Kovatchev, B. (2011). Artificial pancreas: past, present, future. *Diabetes*, 60(11):2672–2682.

[21] Dalton, J. and Deshmane, A. (1991). Artificial neural networks. *IEEE Potentials*, 10(2):33–36.

[22] Delli Priscoli, F., Di Giorgio, A., Lisi, F., Monaco, S., Pietrabissa, A., Celsi, L. R., and Suraci, V. (2017). Multi-agent quality of experience control. *International journal of control, automation, and systems*, 15(2):892–904.

[23] Demers, A., Keshav, S., and Shenker, S. (1989). Analysis and simulation of a fair queueing algorithm. In *Symposium Proceedings on Communications Architectures &Amp; Protocols*, SIGCOMM '89, pages 1–12, New York, NY, USA. ACM.

[24] D.H.Hubel and Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.

[25] Dimitrakopoulos, G. and Demestichas, P. (2010). Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84.

[26] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2014). Long-term recurrent convolutional networks for visual recognition and description. PP.

[27] Doyle, F. J., Huyett, L. M., Lee, J. B., Zisser, H. C., and Dassau, E. (2014). Closed-loop artificial pancreas systems: engineering the algorithms. *Diabetes care*, 37(5):1191–1197.

[28] Duanmu, Z., Ma, K., and Wang, Z. (2017). Quality-of-experience of adaptive video streaming: Exploring the space of adaptations. In *Proceedings of the 2017 ACM on Multimedia Conference*, MM '17, pages 1752–1760, New York, NY, USA. ACM.

[29] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

[30] Eldan, R. and Shamir, O. (2015). The power of depth for feedforward neural networks.

[31] Estan, C., Savage, S., and Varghese, G. (2003). Automatically inferring patterns of resource consumption in network traffic. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03, pages 137–148, New York, NY, USA. ACM.

[32] Fiedler, M., Hossfeld, T., and Tran-Gia, P. (2010). A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36–41.

[33] Fisher, M. E. (1991). A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *IEEE transactions on biomedical engineering*, 38(1):57–61.

[34] Gambuti, R., Canale, S., Facchinei, F., Lanna, A., and Giorgio, A. D. (2015). Electric vehicle trip planning integrating range constraints and charging facilities. pages 472–479.

[35] Gemechu, F., Yu, Z., and Ting, L. (2010). A framework for personalized information retrieval model. pages 500–505.

[36] Goh, A. (1995). Back-propagation neural networks for modeling complex systems. *Artificial Intelligence in Engineering*, 9(3):143 – 151.

[37] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.

[38] Graves, A., Jaitly, N., and Mohamed, A. (2013). Hybrid speech recognition with deep bidirectional lstm. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278.

[39] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4.

[40] "Hansen, P. and Jaumard, B. ("1997"). "cluster analysis and mathematical programming". *"Mathematical Programming"*, "79"("1"):"191–215".

[41] Hardy, W. C. and Preface By-Cardoso, L. (2001). *QoS: measurement and evaluation of telecommunications quality of service*. John Wiley & Sons, Inc.

[42] Hartenstein, H. and Laberteaux, L. P. (2008). A tutorial survey on vehicular ad hoc networks. *IEEE Communications Magazine*, 46(6):164–171.

[43] Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition.

[44] Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

[45] Hemminki, S., Nurmi, P., and Tarkoma, S. (2013). Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 13:1–13:14. ACM.

[46] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. 9:1735–80.

[47] Hou, Z.-S. and Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3 – 35. Data-based Control, Decision, Scheduling and Fault Diagnostics.

[48] Hu, J. and et al. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm.

[49] Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4:1039–1069.

[50] Ibarrola, E., Taboada, I., Ortega, R., et al. (2009). Web qoe evaluation in multi-agent networks: Validation of itu-t g. 1030. In *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*, pages 289–294. IEEE.

[51] J. White, D. (1993). A survey of applications of markov decision processes. 44.

[52] Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201.

[53] Jelassi, S., Rubino, G., Melvin, H., Youssef, H., and Pujolle, G. (2012). Quality of Experience of VoIP Service: A Survey of Assessment Approaches and Open Issues. *Communications Surveys and Tutorials, IEEE Communications Society*, 14(2):491–513.

[54] Joseph, A. D., Beresford, A. R., Bacon, J., Cottingham, D. N., Davies, J. J., Jones, B. D., Guo, H., Guan, W., Lin, Y., Song, H., Iftode, L., Fuchs, S., Lamprecht, B., Kyamakya, K., Fernandez, J. G., Garcia, J. C. Y., Garcia, Y. S. M., de Gracia Santos, J., Nimesh, M., Pan, G., Wu, Z., Wu, Q., Shan, Z., Sun, J., Lu, J., Yang, G., Khan, M. K., and Zhang, J. (2006). Intelligent transportation systems. *IEEE Pervasive Computing*, 5(4):63–67.

[55] Kaelbling, L. P., Littman, M. L., and Moore, A. P. (1996a). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[56] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996b). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

[57] Karim, M. (2017). *Predictive Analytics with TensorFlow: Implement deep learning principles to predict valuable insights using TensorFlow.* Packt Publishing.

[58] KELLER, J. M. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Trans. Syst., Man Cybern.*, 15(4):580–585.

[59] Kok, J. R., Spaan, M. T. J., and Vlassis, N. (2005). Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50(2-3):99–114.

[60] Kropff, J., Del Favero, S., Place, J., Toffanin, C., Visentin, R., Monaro, M., Messori, M., Di Palma, F., Lanzola, G., Farret, A., et al. (2015). 2 month evening and night closed-loop glucose control in patients with type 1 diabetes under free-living conditions: a randomised crossover trial. *The lancet Diabetes & endocrinology*, 3(12):939–947.

[61] Lathia, N., Froehlich, J., and Capra, L. (2010). Mining public transport usage for personalised intelligent transport systems. In *2010 IEEE International Conference on Data Mining*, pages 887–892.

[62] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[63] Lemke, C. and Howson, Jr., J. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423.

[64] Lenz, I., Knepper, R. A., and Saxena, A. (2015). Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*.

[65] Li, L., Lv, Y., and Wang, F. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254.

[66] Liang, X., Du, X., Wang, G., and Han, Z. (2018). Deep reinforcement learning for traffic light control in vehicular networks. *CoRR*, abs/1803.11115.

[67] Littman, M. L. (2001). Value-function reinforcement learning in markov games. *Cogn. Syst. Res.*, 2(1):55–66.

[68] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11 – 26.

[69] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. pages 281–297, Berkeley, Calif. University of California Press.

[70] Mahmood, T. and Ricci, F. (2007). Learning and adaptivity in interactive recommender systems. In *Proceedings of the Ninth International Conference on Electronic Commerce*, ICEC '07, pages 75–84, New York, NY, USA. ACM.

[71] Man, C. D., Micheletto, F., Lv, D., Breton, M., Kovatchev, B., and Cobelli, C. (2014). The uva/padova type 1 diabetes simulator: new features. *Journal of diabetes science and technology*, 8(1):26–34.

[72] Manzoni, V., Maniloff, D., Kloeckl, K., and Ratti, C. (2011). Transportation mode identification and real-time co 2 emission estimation using smartphones how co 2 go works.

[73] Martin, B., Addona, V., Wolfson, J., Adomavicius, G., and Fan, Y. (2017). Methods for real-time prediction of the mode of travel using smartphone-based gps and accelerometer data. *Sensors*, 17(9):2058.

[74] Messori, M., Incremona, G. P., Cobelli, C., and Magni, L. (2018). Individualized model predictive control for the artificial pancreas: In silico evaluation of closed-loop glucose control. *IEEE Control Systems*, 38(1):86–104.

[75] Mitchell, T. (1997). Machine learning.

[76] Moussa, S., Soui, M., and Abed, M. (2013). A multi-criteria decision making approach for personalization itineraries in intelligent transport systems. pages 94–99.

[77] Nagaraja, G. (1990). Applications of a.i. in control systems. In *ACE '90. Proceedings of [XVI Annual Convention and Exhibition of the IEEE In India]*, pages 111–114.

[78] Nuzzolo, A., Comi, A., Crisalli, U., and Rosati, L. (2013). An advanced pre-trip planner with personalized information on transit networks with atis. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2146–2151.

[79] Parker, R. S., Doyle, F. J., and Peppas, N. A. (1999). A model-based algorithm for blood glucose control in type i diabetic patients. *IEEE Transactions on biomedical engineering*, 46(2):148–157.

[80] Pérez-Gandía, C., Facchinetti, A., Sparacino, G., Cobelli, C., Gómez, E., Rigla, M., de Leiva, A., and Hernando, M. (2010). Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring. *Diabetes technology & therapeutics*, 12(1):81–88.

[81] Priscoli, F. D., Fogliati, L., Palo, A., and Pietrabissa, A. (2014). Dynamic class of service mapping for quality of experience control in future networks. In *WTC 2014; World Telecommunications Congress 2014*, pages 1–6. VDE.

[82] Rosenblatt, F. (1957). *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.

[83] Russell, S. J., El-Khatib, F. H., Sinha, M., Magyar, K. L., McKeon, K., Goergen, L. G., Balliro, C., Hillard, M. A., Nathan, D. M., and Damiano, E. R. (2014). Outpatient glycemic control with a bionic pancreas in type 1 diabetes. *New England Journal of Medicine*, 371(4):313–325.

[84] S. Canale, F. Facchinei, R. G. L. P. and Suraci, V. (2014). User profile based quality of experience. *18th International Conference on Circuits, Systems, Communications and Computers (CSCC 2014),Advances in Information Science and Applications*, II.

[85] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[86] Seufert, M., Egger, S., Slanina, M., Zinner, T., Hossfeld, T., and Tran-Gia, P. (2015). A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492.

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

[88] Shaikh, J., Fiedler, M., and Collange, D. (2010). Quality of experience from user and network perspectives. *annals of telecommunications-annales des telecommunications*, 65(1-2):47–57.

[89] Shoham, Y., Powers, R., and Grenager, T. (2003). Multi-agent reinforcement learning: a critical survey.

[90] Solomatine, D. P. (2006). Data-driven modeling and computational intelligence methods in hydrology. *Encyclopedia of hydrological sciences*.

[91] Srinivas, M. and Patnaik, L. M. (1994). Genetic algorithms: A survey. *computer*, 27(6):17–26.

[92] Steil, G. M., Rebrin, K., Darwin, C., Hariri, F., and Saad, M. F. (2006). Feasibility of automating insulin delivery for the treatment of type 1 diabetes. *Diabetes*, 55(12):3344–3350.

[93] Sussman, J. S. (2005). *Perspectives on Intelligent Transportation Systems (ITS)*.

[94] Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9(5):1054–1054.

[95] Swain, P. H. and Hauska, H. (1977). The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics*, 15(3):142–147.

[96] Thabit, H., Lubina-Solomon, A., Stadler, M., Leelarathna, L., Walkinshaw, E., Pernet, A., Allen, J. M., Iqbal, A., Choudhary, P., Kumareswaran, K., et al. (2014). Home use of closed-loop insulin delivery for overnight glucose control in adults with type 1 diabetes: a 4-week, multicentre, randomised crossover study. *The Lancet Diabetes & Endocrinology*, 2(9):701–709.

[97] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Heidelberg.

[98] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103. ACM.

[99] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

[100] Wang, F. Y. (2010). Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications. *IEEE Transactions on Intelligent Transportation Systems*, 11(3):630–638.

[101] Wang, S., Chen, C., and Ma, J. (2010). Accelerometer based transportation mode recognition on mobile phones. In *2010 Asia-Pacific Conference on Wearable Computing Systems*, pages 44–46.

[102] Wang, W., Shi, P., and Basin, M. (2013). Editorial: Special section on data-based control, decision, scheduling and fault diagnostics. *Information Sciences*, 235:1–2.

[103] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1995–2003. JMLR.org.

[104] Ward, E. (1998). *World-class telecommunications service development*. Artech house publishers.

[105] Watkins, C. J. C. H. and Dayan, P. (1992). Technical note q-learning. *Machine Learning*, 8:279–292.

[106] Wei, H., Zheng, G., Yao, H., and Li, Z. (2018). Intellilight: A reinforcement learning approach for intelligent traffic light control. pages 2496–2505.

[107] Weiss, G., editor (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA.

[108] Wilson, D. R. and Martinez, T. R. (1997). Improved heterogeneous distance functions. *J. Artif. Int. Res.*, 6(1):1–34.

[109] Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37 – 52.

[110] Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

[111] Yi, Z., Xiaoguang, Y., Meiping, Y., and Xuemei, Z. (2010). The relationship between urban travelers' personal attributes and propensity to utilize pre-trip traveler information system. pages 1–5.

[112] Zanderigo, F., Sparacino, G., Kovatchev, B., and Cobelli, C. (2007). Glucose prediction algorithms from continuous monitoring data: assessment of accuracy via continuous glucose error-grid analysis.

[113] Zhang, J., Liao, F., Arentze, T., and Timmermans", H. ("2011"a). "a multimodal transport network model for advanced traveler information systems". *"Procedia - Social and Behavioral Sciences"*, "20":"313 – 322".

[114] Zhang, J., Wang, F. Y., Wang, K., Lin, W. H., Xu, X., and Chen, C. (2011b). Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639.