A novel efficient time series deep learning approach using classification, prediction and reinforcement : energy and telecom use case

Aicha Dridi

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAS010

Thèse de doctorat

A novel efficient time series deep Learning Approach using Classification, Prediction and reinforcement: Energy and Telecom use case

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 Ecole Doctorale de l'Institut Polytechnique de Paris (EDIPParis)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Paris, le 28 Novembre 2022, par

AICHA DRIDI

Composition du Jury :

Marcelo Dias de Amorim Directeur de Recherche, NPA Sorbonne Université - LIP6	Rapporteur Président
Enrico Natalizio Professeur des universités, LORIA, Campus Scientifique	Rapporteur
Florence Ossart Professeure des universités, en génie électrique, Sorbonne Université	Examinatrice
Yvon Gourhant R&D at France Telecom	Examinateur
Ghislain Agoua Ingénieur de recherche et Data Scientist chez EDF R&D	Examinateur
Hossam AFIFI Professeur, Télécom SudParis (SAMOVAR)	Directeur de thèse
Hassine Moungla Professeur, Institut Mines-Télécom, Université Paris Descartes	Co-encadrant de thèse
Jordi BADOSA Ingénieur de recherche, École Polytechnique (LMD)	Invité Co-encadrant de thèse

INSTITUT POLYTECHNIQUE DE PARIS

Abstract

Télécom SudParis

Docteur de l'Institut Polytechnique de Paris

A novel efficient time series deep Learning Approach using Classification, Prediction and reinforcement: Energy and Telecom use case

by Aicha DRIDI

A time series is mathematically defined as a series of data indexed by time. It could be seen as a finite sequence (x_1,x_2,... x_n) of length n. Time series cover many real-life phenomena and can be found in many fields. Indeed, it can be a song, the evolution of the electricity consumption of a building or a city, or even the growth of the purchase prices of properties. The work carried out throughout this thesis aims to propose new approaches to treat time series. The first objective is to obtain high-precision predictions. The second is to extract as much information as possible to apply mechanisms such as anomaly detection. Finally, we aim to integrate the obtained results in a tool for optimization and this using deep learning methods.

Indeed, during all the work we have faced several data of diverse nature in the form of time series. This data type is constantly increasing due to the proliferation of tools allowing us to collect and store it.Therefore, time series are at the heart of our thesis.

During the thesis, we dealt with electrical data, cellular data, and Global Positioning System (GPS) taxi tracks.

The electrical data is divided into consumption and production data. The treated consumption data comes from two different buildings. The first building is an incubator. The incubator is divided into 7 zones. For each zone, we have consumption data and the incubator's overall consumption. For the first two zones we also have consumption data categorized by type of consumption (critical, delayable and comfort).

The second building is a student dorm. The data collected from this building comes from the ground floor, floors 1 to 6, the boiler room and overall incoming power. Regarding the production data, it is collected from a photovoltaic panel installation.

The second set of data processed are the cellular traces. They are present in the form of CDR (Call Detail Records) data sets. We dealt with two different CDRs the first CDR contains information on various Base stations in Milan, Italy. The second CDR contains information on various Base stations established in Dakar, Senegal. Every time an interaction between the subscriber and the infrastructure of mobile networks occurs, it is collected and stored in the dataset. The interactions are received/issued calls, SMS, or the Internet. This information is time-stamped based on the interaction.

The third dataset contains traces of taxi mobility in Rome, Italy. It includes the GPS coordinates (Latitude and Longitude) of approximately 320 taxis collected over 30 days. The time information processed is the hour, minute, and second, not the entire date.

The work proposed in this thesis is divided as follows:

First, I reviewed annd analyzed advanced data analysis and artificial intelligence methods in-depth. We chose the more adequate to process the different data categories in our possession. Indeed, with these methods, I could extract the most relevant information.

Since the nature of the three data sets is not the same, we had to adapt the applied processing .

The first studied method was classification since the need for quickly arose. We studied several classification methods based on neural networks to determine the most suitable for our data. For electrical data, we identified two objectives. One was to categorize the type of consumption, and the second was to determine the category of the building according to the consumption habits. For the second dataset, the objective was to determine the daily profile of the base stations. For the third dataset, we didn't apply the classification. There was no need due to the data's nature and the application's future needs.

The second studied method was forecasting. We started by applying several prediction methods on the different datasets. After the various experimentations it was found that the Dense Long Short-Term Memory (DLSTM) provided the best prediction result in terms of performance metrics.

Several predictions have been made, namely, the energy consumption for the two buildings studied during our Ph.D. The prediction of a particular type of consumption. We also varied the forecasted period from the next hour to the next day. In the same way for the production, we predicted the future production of the photovoltaic panels located in the study area with predictions ranging from 1 hour to 24 hours.

Regarding cellular data, we applied several prediction methods to determine the future load of a base station. This information is a very important since it will subsequently allow network operators to adapt their infrastructures to demand. Similarly, to the energy predictions, predicting the load of the base station goes from forecasting the next one hour to the next day.

Dealing with the GPS tracks of taxis in Rome was a little different since the study's purpose was exclusively to predict the next four GPS coordinates of the trajectory to be taken by the taxi.

In the next part of the thesis, we mainly concentrate on the first and second datasets (energy prosumption and cellular data). What makes us mainly concentrate on the energy and cellular data is our intimate conviction that they are highly correlated. Studying the cellular data will allow us to identify the profile of a specific geographical zone. Thus, it will permit us to determine the energy consumption habits of that particular zone since the energy conception is highly dependent on the human presence.

We started by exploring more extensive applications for the processed datasets. We investigated first the semantic compression for the energy data to avoid over-transmission using LoRa networks.

The second investigated method responds to an essential problem which is felt despite the abundance and the increase in data quantity, these data can be erroneous or even lost or suffer from noise. We respond to this problem by proposing using a method called Transfer Learning.

We also explored anomaly detection in cellular networks. We used the result of base station load predictions to develop a mechanism capable of detecting anomalies with an adaptative threshold.

Our last contribution is divided into two parts:

The first aims to develop and implement a solution to manage the consumption of buildings powered by renewable resources (prosumption of renewable energy). The approach integrates the proposed model to perform the prediction in a tool based on deep neural networks and reinforcement learning to allow the dynamic optimization of energy resources (adequacy of supply and demand).

The second allows the deployment of a drone to support a base station in distress. We studied several methodologies present in the literature that we compared to our approach to support the effectiveness of our tool.

For the energy management use case, we started by defining various scenarios. We then created an optimization model for each use case. We resolved the different optimization problems using CPLEX then investigated two heuristic approaches. The first is called Bin Packing and the second is called Rule-Based. The last used approach we examined is a machine learning-based approach. We proposed using two different methods: the first is Q-Learning based and the second is a deep learning approach.

We only studied one use case regarding cellular anomaly management and drone deployment. We defined an optimization model that we resolved. We then used a reinforcement learning approach to deal with drone deployment.

Even if in this work we did not have the opportunity to combine the information related to the processing of the two datasets, we had the opportunity to study each dataset individually.

INSTITUT POLYTECHNIQUE DE PARIS

Résumé

Télécom SudParis

Docteur de l'Institut Polytechnique de Paris

Une nouvelle approche d'apprentissage en profondeur efficace pour le traitement des séries Temporelles utilisant la classification, la prédiction et le renforcement : Cas d'utilisations Energie et télécommunications

par Aicha DRIDI

Une série chronologique est mathématiquement définie comme une série de données indexées par le temps. Il peut être vu comme une suite finie (x_1,x_2,...x_n) de longueur n. Les séries temporelles couvrent un large éventail de phénomènes réels et peuvent être trouvées dans de nombreux domaines. En effet, il peut s'agir d'une chanson, de l'évolution de la consommation électrique d'un immeuble ou d'une ville, ou encore de l'évolution des prix d'achat des biens.

Les travaux menés tout au long de cette thèse visent à proposer de nouvelles approches pour traiter les séries temporelles. Le premier objectif est d'obtenir des prédictions de haute précision. Puis d'extraire le maximum d'informations pour appliquer des mécanismes tels que la détection d'anomalies. Enfin, nous visons à intégrer les résultats obtenus dans un outil d'optimisation et ceci en utilisant des méthodes d'apprentissage profond. Par conséquent, les séries temporelles sont au cœur de notre thèse. En effet, au cours de tous les travaux menés nous avons été confrontés à plusieurs données de nature diverse sous forme de séries temporelles. Ce type de données est en constante augmentation en raison de la multiplication des outils permettant de les collecter et de les stocker.

Le premier jeu de données que nous avons rencontré est formé de données électriques. Ces données sont divisées en données de consommation et de production. Les données de consommation traitées proviennent de deux bâtiments différents. Le premier bâtiment est un incubateur. L'incubateur est divisé en 7 zones. Pour chaque zone, nous disposer des données de consommation et de la consommation globale de l'incubateur. Pour les deux premières zones, nous avons également des données de consommation catégorisés par type de consommation (critique, programmable et confort). Le second bâtiment est un dortoir pour étudiants. Les données recueillies dont les données de consommation du rez-de-chaussée, des étages 1 à 6, de la chaufferie et la puissance globale entrante. Concernant les données de production, elles sont collectées à partir d'une installation de panneaux photovoltaïques. Ces informations sont horodatées avec une granularité de 30 minutes.

Le deuxième ensemble de données traitées sont les traces cellulaires. Ils se présentent sous la forme d'ensembles de données CDR (Call Detail Records). Nous avons traité deux CDR différents, le premier CDR contient des informations sur diverses stations de base situées à Milan, en Italie. Le deuxième CDR contient des informations sur diverses stations de base établies à Dakar, au Sénégal. Chaque fois qu'une interaction entre l'abonné et l'infrastructure des réseaux mobiles se produit, elle est collectée et stockée dans l'ensemble de données. Les interactions sont des appels reçus/émis, des SMS ou Internet. Ces informations sont horodatées en fonction de l'interaction.

Le troisième ensemble de données contient des traces de la mobilité des

taxis à Rome, en Italie. Il comprend les coordonnées GPS (Latitude et Longitude) d'environ 320 taxis collectés sur 30 jours. Les informations temporelles traitées sont l'heure, la minute et la seconde et non la date entière.

Le travail proposé dans cette thèse se décompose comme suit :

Tout d'abord, j'ai fait une étude approfondie des outils avancés d'analyse de données et d'intelligence artificielle. Nous avons choisi le plus adéquat pour traiter les différents types de données qui étaient en notre possession. En effet, avec ces méthodes, j'ai pu extraire les informations les plus pertinentes. La nature des trois jeux de données n'étant pas la même, nous avons dû adapter les traitements appliqués.

La première méthode étudiée est la classification puisque le besoin s'est rapidement fait sentir, nous avons étudié plusieurs méthodes de classification basées sur les réseaux de neurones afin de déterminer la plus adaptée à nos données. Nous avons par la suite appliqué ces méthodes de classification aux deux premiers jeux de données. Pour le troisième ensemble de données, nous n'avons pas appliqué la classification. Cela n'était pas nécessaire en raison de la nature des données et des besoins futurs de l'application.

Pour les données électriques, nous avons identifié deux objectifs. L'un était de catégoriser le type de consommation, et le second était de déterminer la catégorie du bâtiment en fonction des habitudes de consommation. Concernant le deuxième jeu de données, l'objectif était de déterminer le profil quotidien des stations de base.

La deuxième méthode étudiée est la prédiction. Nous avons commencé par appliquer plusieurs méthodes de prédiction sur les différents jeux de données. Après les différentes expérimentations, il a été constaté que Dense Long Short Term Memory (DLSTM) fournissait le meilleur résultat de prédiction en termes de mesures de performance. Plusieurs prédictions ont été faites, à savoir la prédiction de la consommation d'énergie pour les deux bâtiments étudiés au cours de notre doctorat. La prédiction d'un type particulier de consommation. Nous avons également varié la période de prévision qui va de l'heure suivante au jour suivant. De la même manière pour la production, nous avons prédit la production future des panneaux photovoltaïques situés dans la zone d'étude avec des prévisions allant de 1 heure à 24 heures.

Concernant les données cellulaires, nous avons appliqué plusieurs méthodes de prédiction pour déterminer la charge future d'une station de base. Cette information est une mesure très importante puisqu'elle permettra ensuite aux opérateurs de réseaux d'adapter leurs infrastructures à la demande. Comme pour les prévisions d'énergie, la prévision de la charge de la station de base va de la prévision de l'heure suivante au jour suivant.

Le traitement des traces GPS des taxis à Rome était un peu différent puisque le but de l'étude était exclusivement de prédire les quatre prochaines coordonnées GPS de la trajectoire à suivre par le taxi.

Pour ce qui est de la suite de notre travail, nous nous sommes principalement concentrés sur les deux premiers jeux de données (prosomation électrique et cellulaire). Ce qui nous pousse à axer notre étude sur les données énergétiques et cellulaires, c'est notre intime conviction qu'elles sont fortement corrélées. L'étude des données cellulaires nous permettra d'identifier le profil d'une zone géographique précise. Ainsi, cela nous permettra de déterminer les habitudes de consommation énergétique de cette zone particulière puisque la consommation énergétique est fortement dépendante de la présence humaine.

Nous avons ensuite étudié des applications plus étendues pour les jeux de données traités : La première application est la compression sémantique des données d'énergie pour éviter la sur-transmission des données en utilisant le réseau LoRa.

La seconde application étudiée répond à un problème essentiel qui se fait sentir malgré l'abondance et l'augmentation de la quantité de données, ces données peuvent être erronées voire perdues ou souffrir de bruit. Nous répondons à cette problématique en proposant l'utilisation d'une méthode appelée Transfer Learning. La troisième application consiste en la mise en place d'un mécanisme de détection d'anomalie et ceci en utilisant le résultat des prédictions de charge des stations de base et un seuil adaptatif.

Notre dernière contribution est divisée en deux parties:

La première a pour objectif de développer et de mettre en place une solution permettant de gérer la consommation des bâtiments alimentés en ressources renouvelables (prosommation d'énergie renouvelable). La démarche consiste à intégrer le modèle proposé pour effectuer la prédiction dans un outil basé sur les réseaux de neurones profonds et l'apprentissage par renforcement pour permettre l'optimisation dynamique des ressources énergétiques (adéquation de l'offre et de la demande).

La seconde permet le déploiement d'un drone afin de prendre en charge une station de base en détresse. Nous avons étudié plusieurs méthodologies présentes dans la littérature que nous avons comparées à notre approche pour soutenir l'efficacité de notre outil.

Pour le cas d'utilisation de la gestion de l'énergie, nous avons commencé par définir différents scénarios. Nous avons ensuite créé un modèle d'optimisation pour chaque cas d'utilisation. Nous avons résolu les différents problèmes d'optimisation à l'aide de CPLEX puis nous avons étudié deux approches heuristiques. La première s'appelle Bin Packing et la seconde s'appelle Rule-Based. La dernière approche utilisée que nous avons examinée est une approche basée sur l'apprentissage automatique. Nous avons proposé l'utilisation de deux méthodes différentes : la première est basée sur le Q-Learning et la seconde est une approche d'apprentissage en profondeur.

Concernant la gestion des anomalies cellulaires et le déploiement de drones, nous n'avons étudié qu'un seul cas d'utilisation. Nous avons défini un modèle d'optimisation que nous avons résolu. Nous avons ensuite utilisé une approche d'apprentissage par renforcement pour gérer le déploiement de

drones.

Même si dans ce travail nous n'avons pas eu l'opportunité de combiner les informations liées au traitement des deux jeux de données, nous avons eu l'occasion d'étudier en profondeur chaque jeu de données individuellement.

Acknowledgements

I would like to thank the people who have directly or indirectly supported me and contributed to my work during the years of my Ph.D studies.

First, I would like to express my sincere gratitude to my supervisor and thesis director Prof. Hossam Afifi. His knowledge and expertise have guided me through my research. I was blessed to have the opportunity to work and learn with him.

I thank my co-supervisors Dr. Hassine Moungla and Dr. Jordi Badosa for their informed advice motivation, dedication, and patience.

Besides, Iwould like to thank the jury members for accepting my invitation. I am particularly honored by their presence. I thank the reviewers Prof. Marcelo Dias de Amorim and Prof. Enrico Natalazio for their insightful comments and helpful feedback.

I also thank the examiners Prof. Florence Ossart, Ghislain Agoua, and and Yvon Gourhant for their time and flexibility.

My next thanks are addressed to all my colleagues from from Telecom SudParis, and all the colleagues that I met and had to work with for our joint work in our shared office,

Last but certainly not least, I would like to thank my parents, brothers and husband. Thank you for your continued support, and unconditional love. No words can express my appreciation and gratitude. Love you so much.

I would also like to thank my friends ...

This work was supported by the DATAIA Institute, it is a product of research made in collaboration between Télécom SudParis and the Laboratoire de Météorologie Dynamique, and the Laboratoire Génie Electrique et Electronique de Paris

This work benefited from the support of the Energy4Climate Interdisciplinary Center (E4C) of IP Paris and Ecole des Ponts ParisTech. It was supported by 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02].

Contents

Abstract **iii**

Abstract **iii**

Résumé **viii**

Acknowledgements **xiii**

1 General Introduction **1**
1.1 Research issues 1
1.2 Context and problem formulation 4
1.3 Contributions 5
1.4 Structure of the thesis 7

2 Proposed architecture For Time Series Analysis **9**
2.1 Introduction 9
2.2 Time Series Classification 9
2.2.1 Related work 10
2.2.2 Classification Algorithms 10
2.2.3 Performance results for classification 14
2.3 Time Series Forecasting 15
2.3.1 Related work 15
2.3.2 Prediction Algorithms 16
2.3.3 The proposed error functions 20
2.3.4 Hyperprametrisation 21
2.3.5 Prediction algorithms comparison and choice 24
2.3.6 Implementation of native LSTM on ships 25
2.4 Proposed architecture 25
2.4.1 Related work classification aware prediction 25
2.4.2 Classification aware Prediction 25
2.5 Applications 26
2.5.1 Time Series collection 26
2.5.2 Data Colletion architecture 27
2.5.3 Micro Grid 29
2.5.4 Cellular networks 35
2.5.5 Public Transportation 40
2.6 Discussion 42
2.7 Conclusion 42

3 A Deeper Time-series Analysis of the selected applications 45
3.1 Introduction . . . 45
3.2 Semantic compression . . . 45
3.2.1 Introduction . . . 45
3.2.2 Related work . . . 46
3.2.3 Proposed method . . . 47
3.2.4 Performance analysis . . . 48
3.3 From Lack of data to Transfer Learning solution . . . 50
3.3.1 Introduction . . . 50
3.3.2 Related work . . . 50
3.3.3 Transfer learning Architecture . . . 51
3.3.4 Cellular network application . . . 53
3.3.5 Discussion . . . 56
3.4 Anomaly detection . . . 58
3.4.1 Introduction . . . 58
3.4.2 Related work . . . 59
3.4.3 Adaptive Range-based LSTM Prediction Scheme . . . 61
3.4.4 Spatio-Temporal Anomaly Detection Mechanism (STAD) 64
3.5 Conclusion . . . 71

4 Resource Management and optimization Algorithm 75
4.1 Introduction . . . 75
4.1.1 Related work . . . 76
4.2 Energy Use case . . . 79
4.2.1 Context and problem definition . . . 79
4.2.2 Uses cases . . . 82
4.2.3 Exact Resolution . . . 83
4.2.4 Heuristic Resolution . . . 85
4.2.5 Machine Learning Method . . . 88
4.2.6 Performance results for Resource management . . . 93
4.3 Cellular network Use case . . . 100
4.3.1 Related work . . . 101
4.4 Modeling and ILP optimization of the Quality of Service . . . 102
4.4.1 Network Model and architecture . . . 102
4.4.2 ILP Optimization . . . 103
4.5 Reinforcement Learning Approach . . . 105
4.5.1 Results and discussion . . . 108
4.5.2 From Deep LSTM to Transformers . . . 109
4.6 Conclusion . . . 111

5 Conclusion and Perspectives 113
5.1 Conclusions . . . 113
5.2 Perspectives . . . 115

Bibliography 117

List of Figures

1.1 the main functional AI areas 2

2.1 A simple decision tree classifier 12
2.2 Architecture of a three-layer MLP neural network 13
2.3 Convolutional neural network diagram 14
2.4 Efficiency of classification algorithms 15
2.5 One Step Prediction of CDR and Energy using SVR 17
2.6 Recurrent neural network diagram 17
2.7 The LSTM cell 19
2.8 The GRU cell 20
2.9 Multivariate LSTM prediction 23
2.10 UniVariate LSTM prediction 23
2.11 Shift step or prediction window 24
2.12 The proposed architecture 26
2.13 Data collection system actors 28
2.14 One day energy consumption/production 30
2.15 One week energy consumption with classification for two zones 31
2.16 One month energy consumption student dorm, for the 1-6 floors, the ground floor and the heating 31
2.17 One month energy consumption seven zones/total energy consumption 32
2.18 energy consumption prediction with 30 min granularity input data 32
2.19 energy consumption prediction with one hour granularity input data 33
2.20 energy consumption prediction with one hour granularity input data 34
2.21 One day energy consumption/production 35
2.22 Critical energy consumption prediction with one hour granularity input data 35
2.23 Critical energy consumption prediction with one hour granularity input data 36
2.24 Critical energy consumption prediction with one hour granularity input data 36
2.25 Internet load activity from 20^{th} to 30^{th} Nov. in the three cells . 37
2.26 Internet load activity from 20^{th} to 30^{th} Dec. in the three cells . 37
2.27 Classification results visualization over Milan Map 39
2.28 Forecasted trajectories using DLSTM & DGRU 42

3.1 The different steps of the proposed prediction approach 48

3.2 Transfer learning architecture 52
3.3 The pertinence of transfer learning use with One hour granularity prediction 54
3.4 Improvement via transfer learning with 10 minutes granularity prediction 55
3.5 IntraClass Transfer Learning 56
3.6 InterClass TL: downtown to university 57
3.7 InterClass TL: university to night life 58
3.8 System topology 62
3.9 General flow diagram for the algorithm 63
3.10 Network load for storage capacity = 35% and capacity =10%. . 64
3.11 STAD: Spatio-Temporal anomaly detection framework scheme 65
3.12 Duomo square: evolution of numbers of calls) 66
3.13 Examples of Dakar anomalies: the top figure gives an example of a Friday anomaly (red curve) and its previous workdays' normal data (blue curves). The bottom figure shows examples of BS anomalies on February 5th (red curve) and other normal data from Tuesdays. 68
3.14 Comparison between the predicted values of SVR and LSTM for Dakar dataset 69
3.15 Temporal Anomaly detection result for the SanSiro testbed . . 70
3.16 Temporal Friday Anomaly detection results for the Dakar testbed 72

4.1 A Micro Grid Architecture 80
4.2 Microgrid management system Architecture 80
4.3 Bin Packing Problems 86
4.4 RL interaction 91
4.5 First Case CPLEX Resolution for one day 94
4.6 Second Case Rule-Based Resolution for one day 95
4.7 Comparison CPLEX and Rule-Based First Case Resolution . . 95
4.8 Comparison CPLEX and Rule-Based Second Case Resolution . 95
4.9 Performance evaluation of Q-learning algorithm 96
4.10 DL vs RL predicted action for one day: Predicted Use of Battery and Predicted Buy from the grid 98
4.11 DL vs RL predicted action for one day: Predicted Store and Predicted use of solar energy 98
4.12 First case LSTM resolution 99
4.13 Second case LSTM resolution 99
4.14 Third case LSTM resolution 99
4.15 Fourth case LSTM resolution 100
4.16 QoS Management architecture 103
4.17 Finite automate for one day 107
4.18 General flow diagram for the proposed algorithm 107
4.19 Reward and Penalties for 5000 iterations 108
4.20 Reward and Penalties for 5000 iterations 109
4.21 Encoder-decoder sequence to sequence model 111
4.22 Third case LSTM resolution 111

List of Tables

2.1 Metaparameters Choice 21
2.2 Comparisons of prediction methods in terms of processing units 22
2.3 Impact of the shift step on the prediction accuracy. 24
2.4 Efficiency of deep prediction algorithms 24
2.5 Obtained measures for each trajectory for mean square error (MSE) and root mean square error (RMSE) in the test phase. . 41

3.1 Comparison (10 epochs, 3 steps per epoch and 30 neurons) of RNN, LSTM and GRU neural network sizes 49
3.2 Neural network sizes with different configurations 49
3.3 Efficiency of transfer Learning with one hour granularity . . . 54
3.4 Improvement via transfer Learning with 10 minutes granularity 55
3.5 Efficiency of Intra Class Transfer Learning algorithms (university) 56
3.6 InterClass results (downtown to university) 57
3.7 InterClass TL between university and night life classes 58
3.8 Metrics comparison 69
3.9 69

4.1 Variables and definition 82
4.2 Comparison between LSTM and Q-Learning execution Time . 97
4.3 Accuracy of daily predicted actions LSTM vs Q-Learning . . . 97
4.4 Accuracy of daily predicted actions 99
4.5 KPI performance for First case LSTM resolution 99
4.6 KPI performance for Second case LSTM resolution 100
4.7 KPI performance for Third case LSTM resolution 100
4.8 KPI performance for Fourth case LSTM resolution 100
4.9 Notations per cell 104
4.10 Q-Learning execution Time 108
4.11 Q-Learning mean values 109

List of Abbreviations

3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
ACDTW	Adaptive Constrained Dynamic Time Warping
ALSTM	Attention LSTM
API	Application Programming Interfaces
ARFIMA	AutoRegressive Fractionally Integrated Moving Average
ARIMA	Autoregressive Integrated Moving Average
BE	Comfort Energy
BLSTM	Bidirectional LSTM
BP	Bin Packing
BS	Base Station
CE	Critical Energy
CDR	Call Detail Records
CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
CNN	Convolutional Neural Network
CONV	Convolution layer
CPU	Central Processing Unit
D4D	Data for Development
DDOS	Distributed Denial Of Service
DDPG	Deep Deterministic Policy Gradients
DE	Delayed Energy
DER	Distributed Energy Resources
DFTS	Direct Future Time Series forecast
DGRU	Dense GRU
DLSTM	Dense Long Short-Term Memory
DQN	Deep Q-Network
DRL	Deep RL
DTC	Decision Tree Classification
DTW	Dynamic Time Warping
E4C)	Energy4Climate
ECG	ElectroCardioGram
EMS	Energy Management System
ESS	Energy Storage System
FC	Fully Connected
FCN	Fully Convolutional Networks
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit

HVAC	Heating, Ventilation, and Air Conditioning
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IoT	Internet of Things
KNN	K-Nearest Neighbours
KPI	Key Performance Indicators
LMD	Laboratoire de Météorologie Dynamique
LORA	Long Range Radio
LPA	Local Procrustes Analysis
LR	Logistic Regression
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAS	multiagent system
MDP	Markov Decision Process
MG	MicroGrid
ML	Machine Learning
MLP	Multi Layer Perceptron
MPEG	Moving Picture Experts Group
MRI	Magnetic Resonance Imaging
MSE	Mean Square Error
MVNO	Mobile Virtual Network Operators
NB	Naïve Bayes
NSE	Nash–Sutcliffe Efficiency
PG	Policy gradient
POOL	Pooling layer
PPO	Proximal Policy Optimization
PSO	Particle Swarm Optimization
PV	Photovoltaic Panels
QoE	Quality of experience
QoS	Quality of service
RB	Rule Based
ReLU	Rectified Linear Units
ResNet	Residual Networks
RF	Randon Forest
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SMS	Short Message Service
SoC	State of Charge
STAD	Spatio-Temporal Anomaly Detection
SVM	Support Vector Machine
SVR	Support Vector Regression
TATP	Time-Alignment of Time Point forecast
TL	Transfer Learning
TSC	Time Series Classification
URL	Uniform Resource Locator
UTC	Universal Time Coordinated

XML Extensible Markup Language

Chapter 1

General Introduction

1.1 Research issues

It was in the 1940s that the mathematician Norbert Wiener launched the science of the functioning of the human mind, which he called cybernetics. The idea is to model the mind as a black box but this was not conclusive. Two approaches to AI emerged in the 1940s: connectionism and cognitivism. The first aims to reproduce the inner workings of the human brain in a machine. Researchers invent the formal neuron, the first mathematical model of the neuron. For the second, he endorses the hypothesis that thought is analogous to an information processing process. Thought is described there at an abstract level, it is a manipulation of symbols that is independent of the material medium.

It was in 1950 that the notion of artificial intelligence(AI) was born thanks to the mathematician Alan Turing. In his book Computing Machinery and Intelligence, he evokes the possibility of introducing a form of intelligence to machines. Alan Turing devised a test known today as the Turing test to answer his existential question: "Can a machine think". The proposed test aims to estimate a machine's ability to imitate human conversation. To do this, a subject will interact blindly with another human, and with a machine programmed to formulate sensible answers. After 5 minutes of conversation, the man must succeed in determining which of his two interlocutors is the AI.

In this game of imitations, the objective of the machine is not to answer questions correctly, but to answer in the most human way possible by deceiving the human interlocutor by answering. If the subject can determine exactly if it is a machine then the machine has not passed the test otherwise according to Alan Turing the machine can be considered intelligent. Although the relevance of the Turing test is still questioned, researchers still use it.

In 1956, Marvin Lee Minsky, an American scientist, defined artificial intelligence as: "The construction of computer programs which engage in tasks which are, for the time being, performed more satisfactorily by human beings, because they require high-level mental processes such as: perceptual learning, memory organization and critical reasoning."

ELIZA is the first Artificial Intelligence program to pass the Turing test in 1966. This program could analyze a text and search for keywords to respond

coherently to its interlocutor. The program convinced several of its interlocutors that he was a real person. The second IA Parry program passed the test in 1972. This program mimicked the behavior of a paranoid schizophrenic person, and in more than 52% of cases the psychiatrists believed they were talking to a real human being.

AI is made up of several branches that have developed over the years. The following figure 1.1 shows the different branches that make it up. Each branch aims to solve a problem that arises. The beauty of AI is found in the crossing of these branches. For instance, machine learning models can be used to solve problems related to voice, whether it is its generation or transcription.

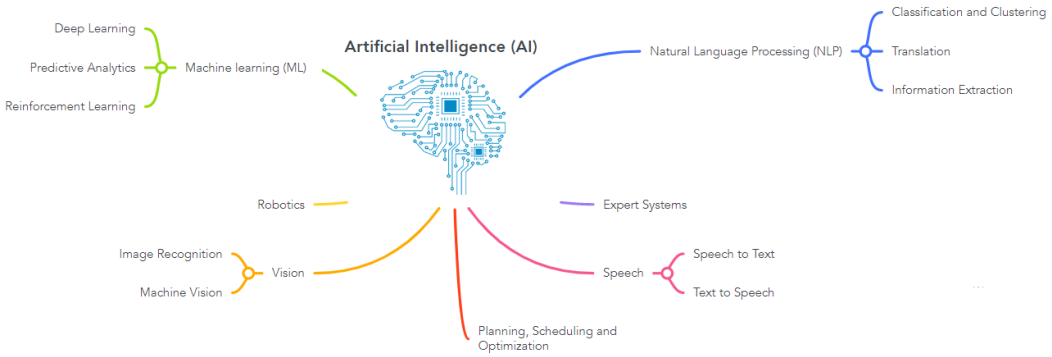


FIGURE 1.1: the main functional AI areas

Despite the rise of AI, and the various promises that have been made such as mass machine translation or even developing a computer capable of beating the world chess champion, in less than ten years. Unfortunately, they could not be held because the computers of the time were unable to store or process information efficiently. These factors have held back the quest for artificial intelligence for several years.

Artificial intelligence has succeeded in taking a new step. Thanks to, the emergence of graphics processors in the 2010s provides the computing power needed to train neural networks. And, the availability of very large correctly annotated databases allowing for finer learning.

Two main approaches to AI:

- The first is symbolic programming, culminating in the 1980s with the development of expert systems. It is a tool capable of reproducing the cognitive mechanisms of an expert in a particular field. The problem is that with such systems, you have to start from scratch when developing a new model: precise and written rules are by nature very difficult or even impossible to generalize from one problem to another (for example, passing from voice recognition to medical diagnosis).

- The second is machine learning which is an approach inspired by the brain. In this case, we program a general model, but the computer adjusts the model's parameters using the data that we provide. This is the most popular approach these days. Some of these models are very

close to statistical methods, but the most famous are inspired by neuroscience: they are called artificial neural networks. In the following, we will mainly focus on this approach. Machine learning relies on mathematical and statistical approaches to allow computers to learn from data, improving their performance in solving tasks without being explicitly programmed for each one.

Machine learning algorithms can be categorized according to the learning mode they employ:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Transfer learning

Over the past few years, science and technology have evolved to finally enable massive parallel computations and expand the horizon of deep learning. Deep learning is a subset of machine learning that employs artificial neural networks to emulate the human brain's learning process.

The main difference between machine learning algorithms and deep learning is that the first one generally needs human correction when they get something wrong. In contrast, the second can improve their results through repetition without human intervention. Deep learning algorithm usually requires big data sets that include various data.

Deep neural networks have proved effective and achieve high accuracy in many application domains. For these reasons, they are one of the most widely employed machine learning methods to solve problems dealing with various data types. AlphaGo [1] program is one of the more famous deep learning algorithms. AlphaGo aims to build a computer Go agent capable of beating the best human player. The inventors of AlphaGo started by presenting the program to several games of Go to learn the mechanics. Then it started playing against different versions of itself thousands of times, learning from its mistakes after each match.

Various research applied deep neural networks for different other applications. It could be used for image recognition. [2], Real-Time Object Detection.

In [3], the authors present YOLO, a new approach to object detection. They frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities.

Another application is natural language processing [4]. In this paper, the author presents a new network architecture, the Transformer, based exclusively on attention mechanisms, dispensing with repetition and convolutions

entirely. They tested the model on an English-to-French translation task and an English-to-German translation task.

Another application is the generative modeling problem [5], where the authors propose a new framework for evaluating productive models via an adversarial process.

In [6], the authors initially formulated time series forecasting problem along with its mathematical fundamentals. Then, they applied the most common deep learning architectures currently successfully predicting time series, highlighting their advantages and limitations. In the rest of our work, we mainly focused on using deep learning to process time series from different fields.

1.2 Context and problem formulation

The massive growth of sensors (Temperature sensor, Humidity Sensor, Accelerometer, Position Sensor) and mobile devices (smartphones, tablets, wearable devices, and mobile phones subscribers) increases the generated data explosively. There will be hundreds of billions of connected devices with the emergence of the Fifth Generation of Cellular Network Technology (5G) [7].

Due to that emergence, the Internet of Things (IoT) applications have been growing and revolutionizing many aspects of human society in the past four decades. These expanding applications also generate big data that can be gathered and managed.

Nevertheless, this rapid growth may impact the existing structures and lead to anomalies. These anomalies may result from different failures depending on the use area.

Considering the Power Grid domain, a broken power substation can sometimes produce sudden energy network blackouts, taking the complete area's power offline. Other types of failures may occur with the grid domain. Significantly, when using alternative power (renewable energy sources), sudden weather changes will impact production and lead to a lack of energy since it is highly dependent on weather conditions.

Sometimes, an over-demand and lousy planning can also create a lack of capacity. Some other failures are due to security issues. Three major blackouts occurred in 2003. The first one was On 14 August 2003 [8]. A blackout happened between Canada and the U.S., affecting approximately 50 million people in eight U.S. states and two Canadian provinces. The second occurred on 23 September 2003 and unfolded in the Swedish/Danish system, and the last arrived in continental Europe on 28 September. This blackout resulted in a complete loss of power throughout Italy.

Depending on the use case failure, different causes could be identified. If we analyse the network failures causes, five leading are pointed out:

- Resource failure may result from an accident and/or natural disaster.
- Hardware failure: A hardware component failure

- Hard Drive Failures: Could result on incompatible Firmware Upgrades or Patches
- Software Failures: A configuration file can go missing or become corrupted, or an expired license
- Security Failures: The most commonly used attack is the Distributed Denial Of Service (DDOS), where the cybercriminals aim to cause an outage

For electrical and cellular networks, overconsumption can lead to an anomaly.

The energy sector is currently facing significant structural changes and challenges: the uses of electricity are constantly increasing (a generalization of air conditioning and electric vehicles), and climate issues require an increase in the share of renewable energies in production (solar and wind). Adapting energy consumption behavior to the available resource is a complex problem since we need to identify consumer habits. The increase in the use of cellular networks allows us to study the human presence in a given place according to a specific base station to predict their behavior and, thus, energy consumption.

1.3 Contributions

The work carried out in this thesis aims to present an architecture capable of processing various time series and interacting with a system based on these time series. Throughout this work, we have used five-time series datasets from three domains: Energy, Telecom, and GPS tracking.

The first dataset we encountered is made up of consumption data. The processed consumption data comes from two different buildings. The first building is an incubator, and the second is a student dormitory.

The second set of data is production data. They are collected from an installation of photovoltaic panels.

The third and fourth datasets processed are cellular traces. They come from CDR (Call Detail Records) data sets. We have dealt with two different CDRs; the first CDR contains information about various base stations in Milan, Italy. The second CDR contains information on various base stations established in Dakar, Senegal.

The fifth dataset contains traces of taxi mobility in Rome, Italy. It includes the GPS coordinates (Latitude and Longitude) of approximately 320 taxis collected over 30 days.

During our thesis, we faced various problems, which we can classify into two main themes.

- Time series processing
- Integration in an optimization tool

We tried to respond to various questions in the first theme:

- How to obtain quality predictions (data not having the same pattern)?
- How to transmit a large volume of data via low-speed networks?
- How to acquire good predictions with an incomplete dataset?
- How to detect an anomaly?

As for the second theme, we mainly tried to answer the following two questions:

- How to make several actors cooperate?
- Which optimization approach to choosing?

To respond to the various issues, we have proposed several contributions.

Our first contribution is proposing a method that allows us to obtain quality predictions. We applied this algorithm to the datasets cited above. To do this, we started by studying the state of the art of the different techniques in detail and compared their results to choose the appropriate solution for our needs. The results obtained pushed us to combine a phase of classification the first time and a stage of prediction the second time.

Secondly, we were interested in more in-depth applications that allow us to exploit the results of our predictions. To respond to how to transmit a large volume of data via low-speed networks? We proposed the semantic compression technique based on the prediction that we applied to our first dataset. Semantic compression aims to find a solution to sending a large volume of data when the transmission support does not allow it.

Then we proposed the use of Transfer Learning. We applied this method to our third dataset to answer two problems the first is the lack of data, and the second is the personalization. This contribution allowed us to respond to the third question, "How to acquire good predictions with an incomplete dataset?".

Anomaly detection is a vast topic. We were interested in detecting anomalies in cellular networks, so we studied it for the cellular domain by exploiting the third and fourth datasets. We first proposed an adaptive threshold and a spatiotemporal method to detect anomalies.

The last contribution is the proposition of a resource management and optimization algorithm. It was applied to two use cases. The first application is the deployment of an energy management system. The second application is applied to the cellular network domain, where we manage the drone deployment in case of an overload on a specific cell which generates the Base Station Failure. We imagined different scenarios we solved using various methods (Bin Packing, Rule-Based, Q-learning, LSTM, and exact resolution).

1.4 Structure of the thesis

This thesis is structured as follow.

Chapter 2 presents the proposed architecture to deal with time series analysis, classification, and forecasting. We show some of the time series collection states of the art. Then we focus on the energy use case by collecting consumption data from the student dorm. After that, we introduced the time series classification with some related work, a comparison between the three main algorithms, and a presentation of some performances. We moved then to Time series forecasting, where we presented the various studied algorithm and how we set the hyperparameters. We finally show in this chapter some results according to the different applications that we made.

Chapter 3 presents some more profound applications to our architecture. We start by investigating the possibility of making semantic compression so that we don't need to send all the collected data. We also examined a technique called Transfer Learning to deal with erroneous or lost data. And finally, we used our prediction results to determine the mean daily load for base stations. This prediction permits us to fix adaptative thresholds to detect eventual anomalies. We also proposed a spatiotemporal method to detect anomalies.

Chapter 4 describes the proposed resource management and optimization algorithm. We first investigate the energy use case. We started by defining multiple scenarios we resolved with exact, heuristic, and machine-learning solutions. We then studied the cellular use case, where we identified a scenario. After that, we fixed it using the exact resolution and two machine-learning methods.

Finally, in the chapter (Chapter 5), we conclude the thesis by providing an overview of our contributions and present possible extensions to the work we realized.

Chapter 2

Proposed architecture For Time Series Analysis

2.1 Introduction

The need for integrating mechanisms for data analysis and exploitation is increasingly felt. The volume of data transferred mandates us to implement an intelligent tool to manage it. Moreover, the existing infrastructures (electrical and cellular) are increasingly suffering from various phenomena (climatic changes, increasing users demands...). Considering the Cellular infrastructure since the development of 3G, 4G, and 5G, the amount of collected and shared data all over the day is increasing, leading to anomalies. The same is true for electrical infrastructures. Since the customers' demand is increasing, an anomaly could lead to an interruption.

In this chapter, we are proposing an architecture for time series analysis. We will go from the collection of the data for the electrical use case to the application of machine learning methods. The applied machine learning techniques presented in this chapter are classification and forecasting. For each process, we will start by exposing a brief state of the art of the existing schemes. We will then disclose the compared algorithms. After that, we will reveal our proposed method with the obtained results. Finally, we will show the obtained results for each method applied to various data sets.

2.2 Time Series Classification

With the development of machine learning, various applications are possible. Classification is one of them. Classification is a general task that can be useful across many subject-matter domains and applications.

In supervised learning, where the used dataset is labeled, the main inference of classification is identifying the category to which a new item belongs based on a training dataset of data containing observations (or instances) whose classes are known. In other words, the goal is to identify a time series coming from one of the possible sources or predefined groups using labeled training data.

It has been shown that classification is accurate to detect not only the carbon footprint produced by household devices [9] but also to detect disease using electrocardiogram data [10].

2.2.1 Related work

Several methods to classify time series data were suggested in the literature. Some research focuses on applying the Randon Forest (RF) to data like MODIS time series. MODIS is a consistent spatial and temporal comparison of vegetation canopy greenness, a composite property of leaf area, chlorophyll, and canopy structure. RF was used in this study to classify crop type, which is essential for food safety and industries [11].

In another work, [12], the authors suggested an architecture called "InceptionTime," a novel deep learning ensemble for time series classification (TSC). The proposed model comprises five distinct Inception networks (deep neural networks) with an architectural design consisting of repeating components referred to as Inception modules. It contains two different residual blocks. Each block comprises three Inception modules rather than traditional fully convolutional layers initialized randomly).

Zhiguang Wang et. al [13] compared different neural network approaches to classify time series. It has been shown that regarding the used benchmarks, the fully convolutional networks (FCN) and the residual networks (ResNet) achieved the best re

In [14], authors proposed and compared the suggested model Long Short Term Memory LSTM-FCNs and Attention LSTM-FCN (ALSTM-FCN) to state-of-the-art models. LSTM-FNC and ALSTM-FCN were tested on all 85 UCR (currently the largest publicly available repository for TSC) time series benchmarks. Results show that LSTM-FCNs can expand FCN models by improving their performance. However, the ALSTM-FCN could not achieve the same performance as the LSTM-FCN on some of the used datasets.

Authors in [15] proposed using a semi-supervised algorithm when the set of labeled examples available is limited. They tested their algorithm on different datasets (ECG Dataset, handwritten the document, Yoga Dataset).

Huanhuan Li and al., [16] proposed an adaptive constrained dynamic time warping (ACDTW) to classify time series. They suggested two penalties to increase the accuracy of the similarity measure within two-time series. The results confirm that ACDTW produces better than four state-of-the-art algorithms on the UCR time series archive.

This section presents some of the investigated classification algorithms in the literature. In the following paragraphs, we will introduce some time series classification algorithms and the methodology chosen to select the algorithm we will use in our architecture. Time series classification has been applied to various domains, agriculture, food safety, handwritten...

2.2.2 Classification Algorithms

In this section, we will introduce two main categories of time series classification algorithms:

- Linear Models
  - Support Vector Machines

  - Logistic Regression

- Non-linear Models
  - K-Nearest Neighbours
  - Naïve Bayes
  - Decision Tree Classification
  - Random Forest Classification

Support Vector Machine (SVM)

Support Vector Machines (SVM) are supervised machine learning models that solve mathematical discrimination and regression problems. The principle of SVMs consists in reducing a classification or discrimination problem to a hyperplane (feature space) in which the data is separated into several classes whose boundary is furthest from the data points. The concept of boundary implies that the information is linearly separable. We propose to use SVM with linear kernels to classify multidimensional time-series datasets. The SVM classification layer is added before the recurrent neural network architecture to improve the deep learning performance model.[17]

Logistic Regression (RL)

It is a powerful supervised Machine Learning (ML) algorithm used for binary classification problems (when the target is categorical) [18]. It is a statistical model for studying the relationships between a set of qualitative variables, Xi, and a qualitative variable, Y. It is a generalized linear model using a logistic function as a link function[19]. RL can be used for classification and class probability estimation because it is linked to logistic data distribution. It takes a linear combination of features and applies a nonlinear sigmoidal function to them. The most commonly used approach is to have one binary output variable. However, we can observe in some cases multiple classes outputs (multinomial logistic regression) [20].

K-Nearest Neighbours (KNN)

kNN is a standard classification algorithm that relies exclusively on the choice of the classification metric. It is "non-parametric" (only k must be fixed) and is based only on training data [21]. It is a distance-based classifier, and the metric used to determine class membership is the Euclidean distance. To estimate the output associated with a new input x, the k nearest neighbors method considers (in an identical way) the k training samples whose information is closest to the new input x, according to a distance to be defined. KNN can be adapted for time series using the dynamic time warping (DTW) metric distance [22].

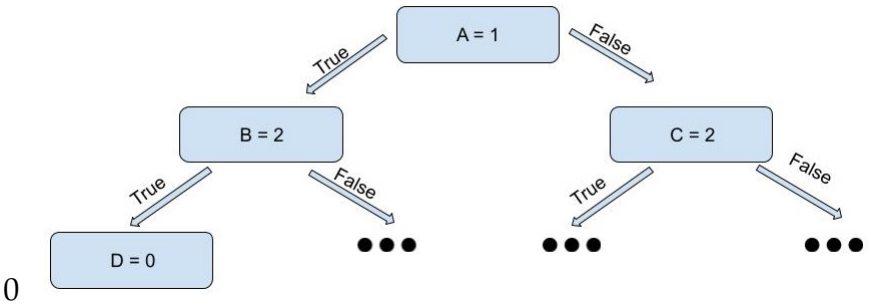
Naïve Bayes (NB)

Bayes' theorem is based on conditional probabilities: "What is the probability that an event will occur knowing that another event has already occurred." [23] In real applications of Naive Bayes, the Outcome is calculated based on several variables that make the calculation complex. To avoid making complex calculations, one approach is to consider these variables independently. This is a strong assumption. Generally, the predictor variables are related to each other. The term "naive" comes from the fact that we assume this independence of the variables [24].

Naive Bayes classifiers are well used in many real-world situations, such as document classification and spam filtering.

Decision Tree Classification (DTC)

DTC is used as a Supervised Learning technique in both continuous and categorical problems [25]. It is a decision support tool representing a set of choices in the graphical form of a tree. The different possible decisions are located at the branches' ends (the tree's " leaves ") and are reached according to decisions made at each stage. Given data of attributes and their classes, a decision tree produces a sequence of rules that can be used to classify the data [26].



0

FIGURE 2.1: A simple decision tree classifier

Random Forest Classification

The "random forest" algorithm was proposed by Leo Breiman and Adèle Cutler in 2001 [27]. In its most classic form, it performs parallel learning on multiple randomly constructed decision trees trained on different subsets of data. The ideal number of trees, which can go up to several hundred or even more, is an important parameter: it is very variable and depends on the problem. Concretely, each tree of the random forest is trained on a random subset of data according to the principle of bagging, with a random subset of features (variable characteristics of the data) according to the principle of "random projections." The predictions are then averaged when the data is quantitative or used for voting for qualitative data, in the case of classification trees [28]. RF requires little data pre-processing.

Multi Layer Perceptron (MLP)

MLP is a type of feed-forward artificial neural network. They are neural network models that work as universal approximators. They can approximate any continuous function. MLPs are formed of neurons called perceptions. A perceptron is a single-neuron model precursor to more extensive neural networks. A perceptron receives n features as input (x = x_1,x_2,...x_n), and each element is associated with a weight. Input features must be numeric. So, non-numeric input features must be converted to numeric ones to use a perceptron [29]. MLP combines several neurons into at least three layers: an input, hidden, and output layer. The input layer receives the input signal to be processed. The output layer performs the prediction or classification tasks. The real computational engine of the MLP is the arbitrary number of hidden layers placed between the input and output layers [30]. MLP is a supervised learning algorithm that is based on back-propagation techniques.

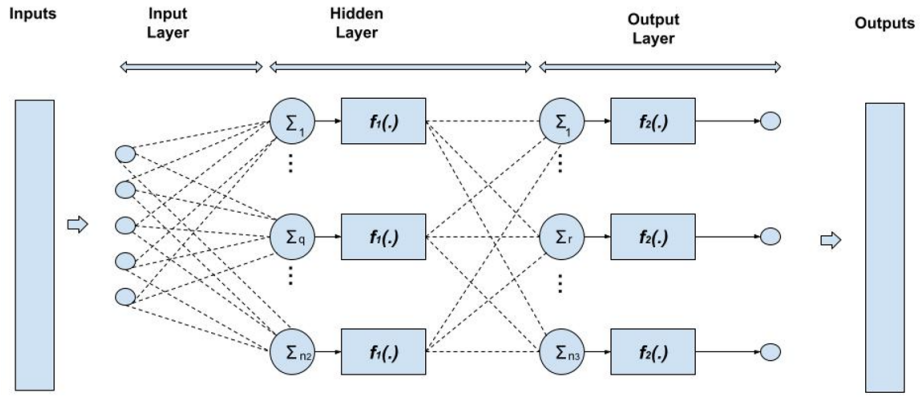


FIGURE 2.2: Architecture of a three-layer MLP neural network

Convolutional Neural Network (CNN)

CNCNN is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the picture, and be able to differentiate one from the other [31]. Generally, the architecture of a Convolutional Neural Network is composed of [32]:

- **Convolution layer (CONV):** The role of this first layer is to analyze the images provided as input and to detect the presence of a set of features.
- **Pooling layer (POOL):** receives as input the feature maps formed at the output of the convolution layer, and its role is to reduce the size of the images while preserving their most essential characteristics.
- **The ReLU (Rectified Linear Units) activation layer:** This layer replaces all negative values received as inputs with zeros.

- **Fully Connected (FC) layer:** These layers are placed at the end of the CNN architecture and are fully connected to all output neurons. After receiving an input vector, the FC layer successively applies a linear combination and then an activation function with the final aim of classifying the input image. Finally, it returns as output a vector of size d corresponding to the number of classes in which each component represents the probability for the image input to belong to a class.

Fig 2.3 shows the different layers with the interaction between them.

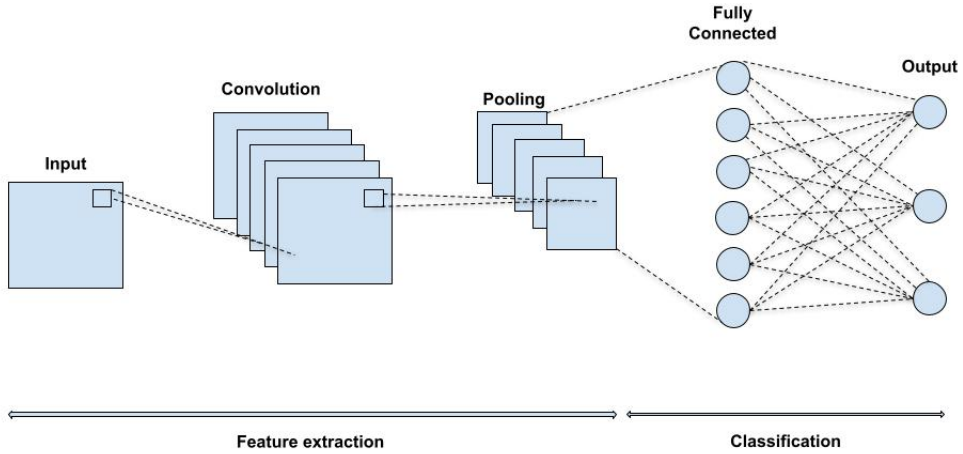


FIGURE 2.3: Convolutional neural network diagram

2.2.3 Performance results for classification

Classification uses two years of data for different energy classes. As it can be deduced from Fig. 2.4, we have compared the classification efficiency of a convolutional neural network, multilayer perceptron, and support vector machines [33]. We can see that the MLP outperforms the two other methods. One could think that CNN would give better results, but since it needs much more data to converge, training stops too early before reaching a good tuning of its weights.

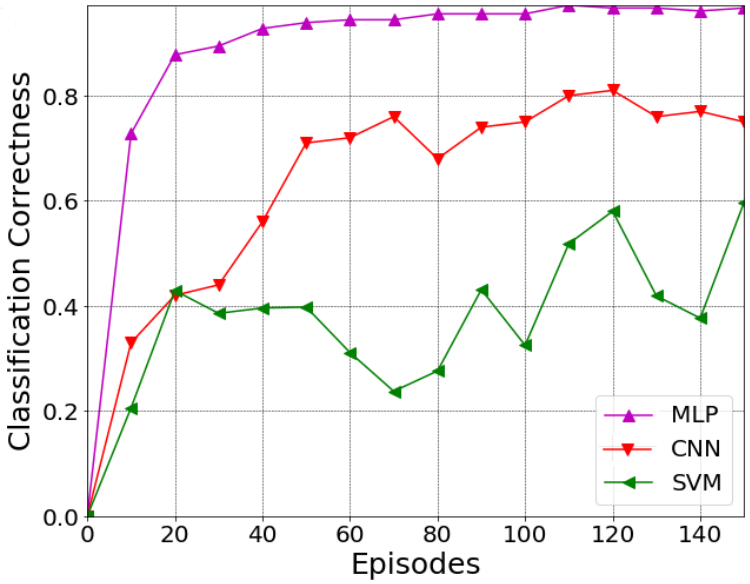


FIGURE 2.4: Efficiency of classification algorithms

2.3 Time Series Forecasting

Time series prediction is a relatively mature technology, which has already proven itself since it dates back to the 80s (even if there are more recent techniques based on deep learning). Using current and historical data helps create hypotheses and predictions about future events. The forecast allows for the real-time alignment of supply and demand but also applies a posteriori as a management tool. Prediction or forecasting can be helpful across many domains and applications.

2.3.1 Related work

We find various approaches to deal with time series prediction in the literature. Using statistical algorithms, authors in [34] applied naïve and random walk AutoRegressive Fractionally Integrated Moving Average (ARFIMA) and Exponential Smoothing Method + Box-Cox Transformation + ARMA (BATS) methods to predict the monthly temperature and rain precipitation. Results show that ARFIMA and BATS models are the most accurate considering Root Mean Square Error (RMSE) and Nash–Sutcliffe Efficiency (NSE).

In [35], the author used Support Vector Regression (SVR) to predict the future load of a Base Station (BS) so that we can dynamically detect anomalies. The objective is to decide when to allocate/free resources.

AKDI et al. [36] compared two different neural network paradigms: The autoregressive integrated moving average (ARIMA) and the harmonic regression model, to predict the daily electrical energy consumption. If the data include a periodic component, the harmonic regression model will give better results.

In [37], authors used two Recurrent Neural Networks (RNN): time-alignment of time point forecast (TATP) and direct future time series forecast (DFTS) to

predict prices of agricultural products. The result shows that TATP has better accuracy.

[38] use the building's characteristics, monthly electricity, and gas bills to predict energy consumption.

In another work [39], authors try to find the impact of human occupancy (presence) on energy consumption behaviors.

In [40], authors proposed a forecasting model using the Bayesian network and dynamic programming principles.

Cao et al. [41] combined the complete ensemble empirical mode decomposition with adaptive noise (CEEMDAN) with the Long Short Term Memory (LSTM) to predict the stock market price. They compared their model with several models, including LSTM, Support Vector Machine (SVM), CEEMDAN-SVM, CEEMDAN-Multi-Layer Perceptron (MLP), and CEEMDAN-LSTM on the same dataset. They concluded that the proposed model CEEMDAN-LSTM outperformed the proposed model according to the following error measures Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). These performance measures are the most widely used to identify and compare the reliability of such algorithms, and hence they will be used hereafter.

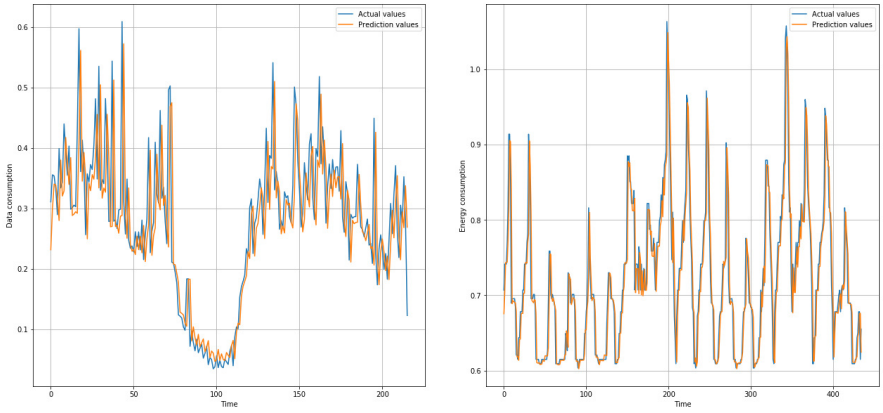
It is important to note that classification and prediction are usually applied together. It is essential to recall that both techniques are necessary for any general wide-scale time series analysis tool.

2.3.2 Prediction Algorithms

Support Vector Regression (SVR)

SVR is a compoment of SVM. The Support Vector Regression (SVR) uses the same ideas as the SVM for classification, with a few differences. For SVR, a result is an actual number. The nature of the output made it difficult to forecast the information, which has infinite possibilities. A margin of error (epsilon) is defined in the case of regression as an approximate estimate for the SVM. SVR is more complex. It is considered a baseline for prediction. Support Vector Regression (SVR) is an effective machine learning technique that can be used in time-series analyses [17]. It is among the best "off-the-shelf" supervised learning algorithms.

We show in Fig. 2.5 the actual and the prediction values in two different time series contexts. On the left, we offer the application of SVR on 5G cellular data, while on the right, we apply the same algorithm to the energy consumption dataset.



(A) CDR prediction (B) Energy consumption prediction

FIGURE 2.5: One Step Prediction of CDR and Energy using SVR

Recurrent Neural Networks

Various research works have been carried out on forecasting time series in diverse areas. RNN is a class of artificial neural networks where each neuron in the network is sequentially connected to a predecessor, and a follower [42].

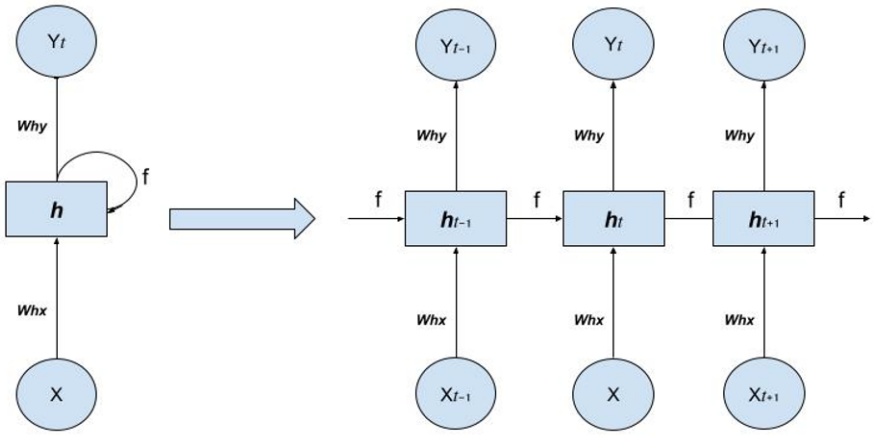


FIGURE 2.6: Recurrent neural network diagram

Fig 2.6 shows a one-unit recurrent neural network connecting the input and output of the network. On the right is the "unfolded" version of the structure. RNN is a class of neural networks that can 'memorize' sequences based on history [43, 44]. RNN formulation could be of the following form: Let $X = (x_1, ..., x_t)$ be the input sequence, $Y = (y_1, ..., y_t)$ the output vector sequence and hidden state of the memory cell. $H = (h_1, ..., h_t)$.

$$h_t = H(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{2.1}$$

$$p_t = W_{hy}y_{t-1} + b_y \tag{2.2}$$

Where:
H is a nonlinear activation function that keeps neuron output bound to a specific interval. W_{hx},W_{hy}, W_{hh} respectively corresponds to the weight between

the input and hidden layer, the hidden and output layer and hidden layers, b_y and b_h symbolize the bias vectors for the output and hidden layers.

Long Short-Term Memory (LSTM)

Even if RNN generates excellent results, it suffers from a problem caused by the retro-propagation in time, also called the vanishing gradient problem. LSTM was proposed in 1997 to surmount this problem [45]. The LSTM comprises three layers: Input, Hidden, and Output Layer [46]. It is constituted of memory blocks with self-loops. Each memory block is composed of special multiplicative units called gates.

The memory blocks handle the flow of information within the network with the help of three gates. The input and forget gates both work on the state of cells. The role of the input gate is to selectively record new information into the cell state. At the same time, forget-gate is aimed at selectively forgetting information that is no longer required for LSTM understanding. The output gate is responsible for picking helpful information about the current cell state and dispensing it as an output.

The LSTM learns to keep only pertinent information to make predictions. This is achieved during the retro-propagation (training phase) [47]. Figure 2.7 represents the structure of LSTM blocks. A single LSTM cell's operation is represented by equations 2.3 to 2.8.

$$i_t = \sigma(W_{ix}x_t + W_{hh}h_{t-1} + W_{ic}c_{t-1} + b_i) \tag{2.3}$$

$$f_t = \sigma(W_{fx}x_t + W_{hh}h_{t-1} + W_{fc}c_{t-1} + b_f) \tag{2.4}$$

$$o_t = \sigma(W_{ox}x_t + W_{hh}h_{t-1} + W_{oc}c_{t-1} + b_o) \tag{2.5}$$

$$u = tanh(W_{ux}x_t + W_{hh}h_{t-1} + W_{u}c_{t-1} + b_u) \tag{2.6}$$

$$c_t = f_t * c_{t-1} + i_t * g(W_{cx}x_t + W_{hh}h_{t-1} + W_{cc}c_{t-1} + b_c) \tag{2.7}$$

$$h_t = o_t * h(c_t) \tag{2.8}$$

Where:

$$\sigma(x) = \frac{1}{1+e^x} \tag{2.9}$$

Equations are essential to understand retro-propagation. The i_t, f_t, o_t, u, c_t, σ respectively correspond to the input gates, the forget gates, the output gates, the update signal, the memory cells, and the sigmoid activation function. Output is derived concerning the expected value (label). The Weights are recalculated until the error is minimized (gradient descent).

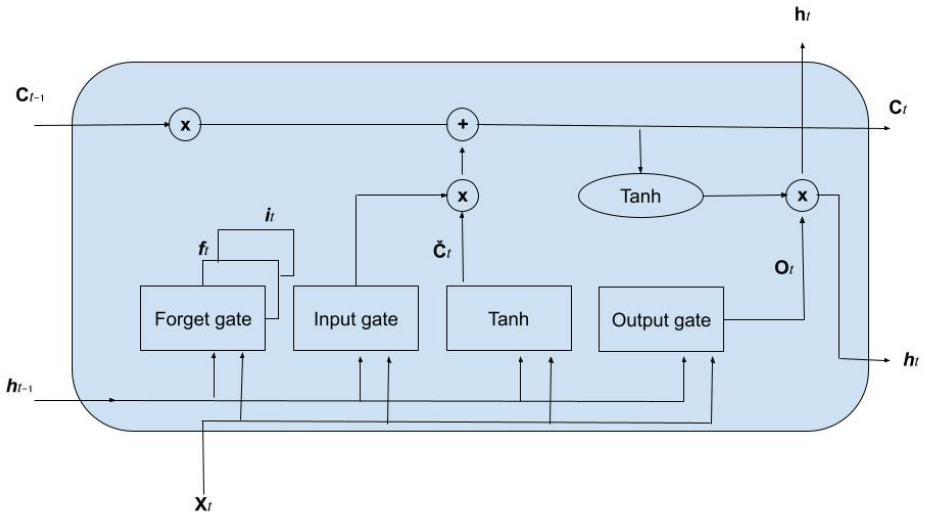


FIGURE 2.7: The LSTM cell

We can find in the literature different variants of the LSTM. We can cite the following:

Multiplicative LSTM : It was introduced in [48] as a recurrent neural network architecture for sequence modeling that combines the long short-term memory (LSTM) and multiplicative recurrent neural network (mRNN) architectures. The mRNN and LSTM architectures can be connected by adding connections from the mRNN's intermediate state m_t to each gating unit in the LSTM. Since then, the mLSTM has been categorized as a high-profile, state-of-the-art achievement in natural language processing.

LSTMs With Attention : Attention is the idea of freeing the encoder-decoder architecture from the fixed-length internal representation [49]. This is achieved by keeping the intermediate outputs from the encoder LSTM from each step of the input sequence and training the model to learn to pay selective attention to these inputs and relate them to items in the output sequence.

Bidirectional Long Short-Term Memory (BLSTM): Bidirectional LSTM is just putting two independent RNNs together [50]. This structure allows the networks to have both backward and forward information about the sequence at every time step.

Dense LSTM : DLSTM is an LSTM network to which we added a dense layer. The dense layer is where each neuron is connected to all the neurons from the next layer [51].

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is another implementation of RNN. It was proposed in 2014 by Cho et al.[52]. The main difference between GRU and LSTM is that it's easier to implement and compute. The GRU is constituted of two gates. An update gate z decides how much the block updates its activation or content. Similarly to the forget gate on LSTM, a reset gate r allows a block to ignore the previously computed state.

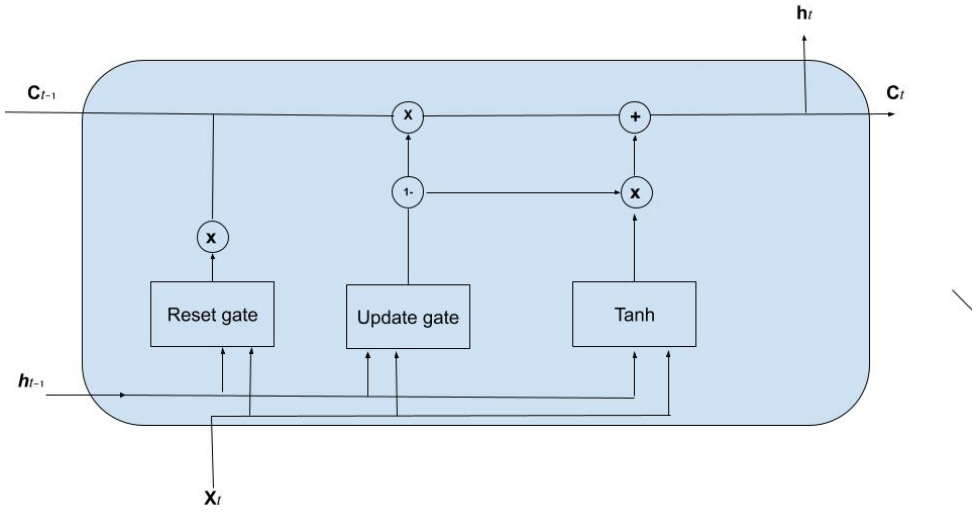


FIGURE 2.8: The GRU cell

A single GRU cell's operation is represented by equations 2.10 to 2.13

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{2.10}$$

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{2.11}$$

$$\widetilde{h}_t = tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \tag{2.12}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{2.13}$$

The z_t, r_t, $\widetilde{h}_t$, tanh, h_t, respectively, correspond to the update gate, reset gates, candidate hidden layer values, activation function, and hidden layer values at time t.

2.3.3 The proposed error functions

Error function calculations are made in two different phases. The first one is the back-propagation evaluation produced while we train the neural network. The NN compares the output value to the actual output that we want to reach (called the label) and backpropagate this error from the last cell to the first one by a set of recursive mathematical equations.

Once the training phase is done, we pass to the evaluation phase or testing phase to observe the efficiency of our neural network prediction. During this step, we will use the error function described in equations 2.14 to 2.16 that

gives the respective formulas for RMSE, MSE, and R^2. These formulas are going to be used to evaluate our proposed prediction solution.

$$RMSE=(\frac{1}{n}\sum_{i=1}^{n}(P_i-\hat{P}_i)^2)^{1/2} \tag{2.14}$$

$$MSE=\frac{1}{n}\sum_{i=1}^{n}(P_i-\hat{P}_i)^2 \tag{2.15}$$

$$R^2=1-\frac{SSE}{SSI} \tag{2.16}$$

Where:

$$SSE=\sum_{i=1}^{n}(P_i-\hat{P}_i)^2 \tag{2.17}$$

and,

$$SST=\sum_{i=1}^{n}(P_i-\bar{P}_i)^2 \tag{2.18}$$

Where:
P_i, $\hat{P}_i$, $\bar{P}_i$, n, are respectively the actual, predicted, mean value and the number of samples.

2.3.4 Hyperprametrisation

Before training a neural network, the first step is determining the best meta parameters to apply to the neural network during the learning phase. Meta parameters are the number of neurons, the number of the hidden layer, the number of iterations, and the batch size... We trained our model using different inputs to choose the most accurate ones related to our predictions.

The results were evaluated using metrics such as Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE).

We varied the batch size, neuron number, number of layers, and number of iterations. We then compared the values of error metrics for each configuration. Table 2.1 shows the steps of choosing the best meta parameters for our learning algorithms. For each variable to predict, we varied the different meta parameters to determine the most accurate for making the optimal prédictions.

TABLE 2.1: Metaparameters Choice

Config	batch size	N°neurons	N° layers	N° iterations	MSE	RMSE	R2
1	64	100	6	200	0.011	0.105	10.690
2	64	100	6	4000	0.003	0.057	5.582
3	100	100	4	4000	0.006	0.076	0.129
4	100	100	6	4000	0.003	0.053	0.493
5	100	100	6	6000	0.003	0.056	0.497
6	256	100	6	4000	0.004	0.057	0.361

CPU vs GPU

In this section, we evaluate the processing acceleration depending on whether it is running on the CPU or GPU.

TABLE 2.2: Comparisons of prediction methods in terms of processing units

Hardware \ Approach	GRU	LSTM	SVR
CPU	1418.81s	1288.62s	17.75s
GPU	853.99s	820.41s	16.42s

In Tab. 2.2, we compare the above prediction methods in terms of computational load. We see that the GPU-enabled hardware accelerates processing of deep recurrent neural networks such as LSTM and GRU architectures.

Uni-variate and multi-variate Prediction

Time series prediction can be classified into two categories: univariate and multivariate. It corresponds to a scenario with one or many inputs and one output or many inputs and many outputs. As an example of univariate, one could feed the input type with inputs such as hour, day, temperature, and the output as the daily energy consumption. For example, for multivariate, we can imagine that we want to predict several outcomes such as daily energy consumption, human presence, and renewable energy production. Il will be shown in the next part that uni-variate (many to one) labeling gives better prediction results for preciseness.

The original dataset is a multivariate time series that contains multiple features. The features are decomposed into seven zones. In this case, we are considering the following four input features ($Zone1\_red$, $Zone1\_yellow$, $Zone1\_green$, and $Air\ Temperature(C)$). Each *color* represents a consumption type (critic, delayed, and comfort consumption).

The considered input features are enhanced by the day of the week, the hour of the day, and the minute of the hour to have an accurate prediction. We show how these features vary across time in Fig. 2.9. Then, we adopted the LSTM forecasting approach with a multi-step prediction model setup since it outperforms the abovementioned methods. The model learns to predict a range of future values (many to many) from each consumption type. Our model's training data consists of observations over the past 24 hours. Then the model learns to predict the next 24 hours for the three types of consumption.

In Fig. 2.10, we have restricted the number of features in the original dataset to one. In the following figures, we only predict the $Zone1\_red$ consumption (a building zone with only the critic energy consumption). The result shows that LSTM reports better accuracy with univariate time series than multivariate time series.

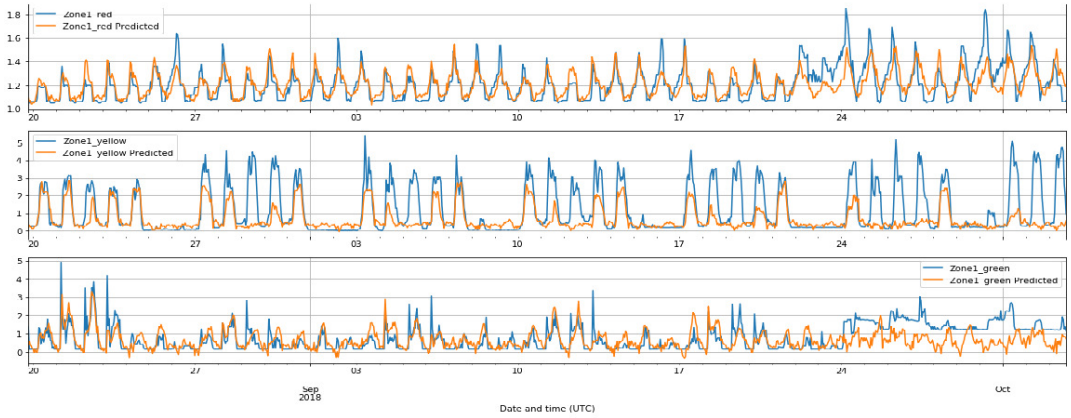


FIGURE 2.9: Multivariate LSTM prediction

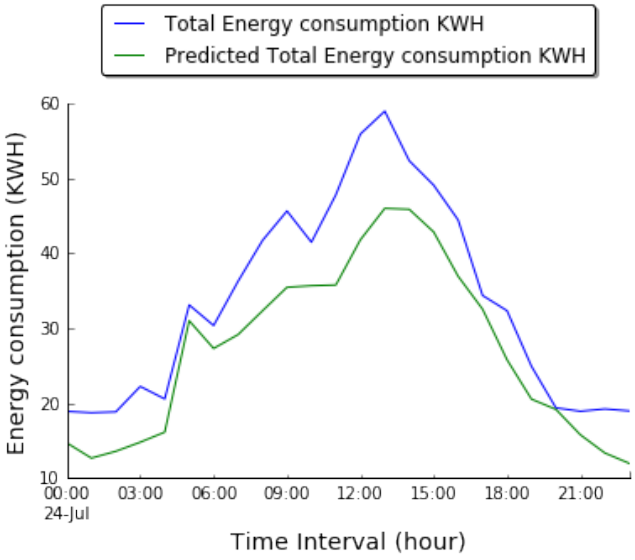


FIGURE 2.10: UniVariate LSTM prediction

Varying the shift step

Time series prediction uses RNN techniques to predict future events. Events are univariate/multivariate observations that vary over time. Observations (e.g., in the MicroGrid dataset) are recorded every 60 minutes. This means that a single day will contain 24 observations.

Time series prediction interval affects the prediction precision. Logically, the larger the prediction interval is, the less precise it will be. During the data processing, we define the desired number of observations to predict. In other words, the shift step is the prediction window or the number of observations we aim to predict. Fig. 2.11 details how we deal with the input and labels.

Moreover, we have assessed the impact of the number of target values (steps) that can be predicted. In the following table 2.3, we show the varied-steps prediction using LSTM and the impacts on the chosen KPI. Results show that a single-step prediction outperforms the multi-step prediction model. In the single-step setup, the LSTM model learns to predict a single point (not

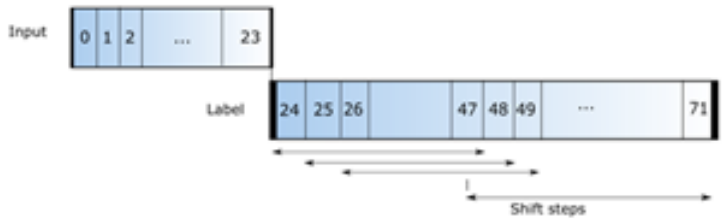


FIGURE 2.11: Shift step or prediction window

a sequence as before) in the future based on some history provided. The considered data set is composed of different entries. It describes the energy consumption of a building divided into different zones. In this dataset, energy consumption is classified into critical, delayed, and comfort.

TABLE 2.3: Impact of the shift step on the prediction accuracy.

Consumption / Shift	MSE	RMSE	MAE	MAPE	R2
Critical Consumption 1 hour	0.004	0.062	0.043	6.276	0.361
Critical Consumption 24 hours	0.004	0.065	0.047	6.799	0.290
Delayable Consumption 1 hour	0.027	0.164	0.099	inf	0.437
Delayable Consumption 24 hours	0.039	0.197	0.117	inf	0.226
Comfort Consumption 1 hour	0.002	0.048	0.034	393.658	0.106
Comfort Consumption 24 hours	0.039	0.197	0.117	inf	0.226

2.3.5 Prediction algorithms comparison and choice

In Tab. 3.3, we compare the prediction accuracy of the time series forecasting approaches (LSTM, GRU, and SVR) in terms of different KPIs such as Mean Squared Errors (MSE), Root MSE, and R^2. Results show that LSTM outperforms GRU and SVR, while the latter is still valid for small and medium datasets.

TABLE 2.4: Efficiency of deep prediction algorithms

Approach / KPI	GRU	LSTM	SVR
MSE	0.007	0.006	0.0175
RMSE	0.076	0.063	0.073
R^2	0.911	0.918	-10.1753

2.3.6 Implementation of native LSTM on ships

In a collaborative work, [53], an LSTM algorithm was embarked onto an electronic device. The devices used for this experiment are STM32L476 [54] as measurement and calculation device, which generates prediction data and compares it to measurements, Semtech's SX1276MB1MAS [55] as LoRa transmission board, and STM32 Nucleo Expansion Board [56] to measure the energy consumption of LoRaWAN transmissions.

The data sets we used are occupancy data of cellular base stations for the 1st data set and power consumption of a smart building for the 2nd one, in the function of the time. The LSTM network was implemented in C and embarked the LSTM network directly on the sensor.

2.4 Proposed architecture

Since in our work, we are dealing with various time series from different applications, and we need to differentiate between them, we decided to combine the classification methods and the prediction one in our proposed tool. As it will be shown, we believe that combining a classification and prediction model minimizes the squared errors, which leads to more accurate predictions. Moreover, we have shown that using machine learning prediction to make multivariate prediction leads to worse results.

2.4.1 Related work classification aware prediction

Wang et al. [57] applied the CNN-LSTM to analyze emotions using input text. Tae-Young Ki and Sung-Bae Cho [58] used a similar approach to forecasting energy consumption. They compared the proposed method with linear regression model (LR), random forest (RF), decision tree (DT), regression, and multilayer perceptron (MLP). The proposed method achieved the lowest MSE score by 10_fold cross-validation experiments.

Ullah et al. [59] used CNN models with BLSTM and applied them to video processing, particularly to recognize human actions. The author in, [60] The authors have reviewed some deep learning methods and the work done with them. They also confirm that employing deep learning in the time-series analysis is better than the previously existing statistical techniques.

This chapter compares the use of prediction algorithms for time series and classification algorithms for time series. Also, Oh et al.[61] combined CNN and LSTM and applied them to the medical field to accurately detect arrhythmia in the ECG. We decided to combine those methods so that we could obtain better results.

2.4.2 Classification aware Prediction

As it was justified in the performance section, our architecture is based on MLP and LSTM. The combination of MLP and LSTM is very powerful in

leveraging the Spatiotemporal features (the type of consumption and the target values) for prediction.

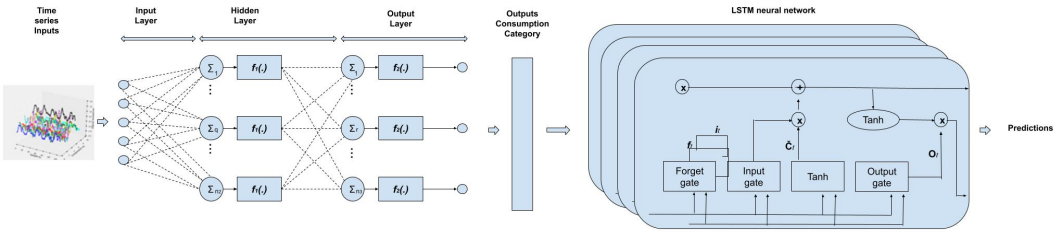


FIGURE 2.12: The proposed architecture

2.5 Applications

2.5.1 Time Series collection

Time series data are one of the most common types of collected data. It is a collection of observations obtained through repeated measurements over time. To track changes over time, we use these successive measurements. Time series data are also known as time-stamped data. A time series can be taken on any variable that changes over time. It is everywhere since time is a constituent of everything that is observable.

Moreover, with the increase in the deployment of measurement tools, sensors and systems are constantly emitting a continuous stream of time series data. This has generated new needs like the need to store this data, the need to store compressed data without losing information, the need to offer a better quality of service/experience (QoS/QoE), and many others... The proliferation of data provides new opportunities, to name some of them:

- Tracking hourly, daily, or weekly weather data
- Tracking the energy produced with renewable sources
- Tracking the energy consumption
- Traking the mobility
- Tracking the intrusion detection
- Tracking changes in application performance
- Tracking the trend in financial markets

With the different applications that time series offer, we can identify factors influencing certain variables from period to period. We can also see how a given load, security, or monetary variable changes over time. We could also use classification and forecasting methods.

Our thesis is part of the Peper project, which aims to offer the microgrid an efficient energy management system. The efficient management of a microgrid is based on predicting the behavior of the different actors: producers and consumers. We can thus envision a system where the equilibrium in real-time is no longer provided only by the modulation of production but also by the adjustment of the demand, thus making the consumer a central actor.

Therefore, the project's objective is to gather data on the different actors, exploit learning techniques and Deep Learning to develop classification and forecasting algorithms for production and consumption and establish collaboration between the different actors. In other words, the goal is to find a way to make production, consumption, and storage cooperate to use renewable energies better.

The success of this project depends on data collection: indeed, it is essential to collect electricity consumption, electricity production, and meteorological data since renewable production strongly depends on it.

The Laboratoire de Météorologie Dynamique (LMD) provides us with consumption data from a startup incubator in Ecole Polytechnique and meteorological data and production information from photovoltaic panel installations. Since it was essential for the project to collect consumption data from different building types, an electricity consumption collection system in a student dorm was carried out.

Energy consumption monitoring precisely measures the consumption of different electric devices at different scales. It goes from high-power industrial plants to small buildings. The goal is two-fold:

- Cyclic monitoring for potential energy consumption analysis, management, and reduction
- Outlier early detection

2.5.2 Data Colletion architecture

In this subsection, we will describe the different elements that interact to collect energy consumption information from the student dorm and build a scalable data-sensing architecture. Fig. 2.13 describes the data collection system. An energy monitoring and supervision tool were installed in a student dorm. The chosen solution is EGauge [62].

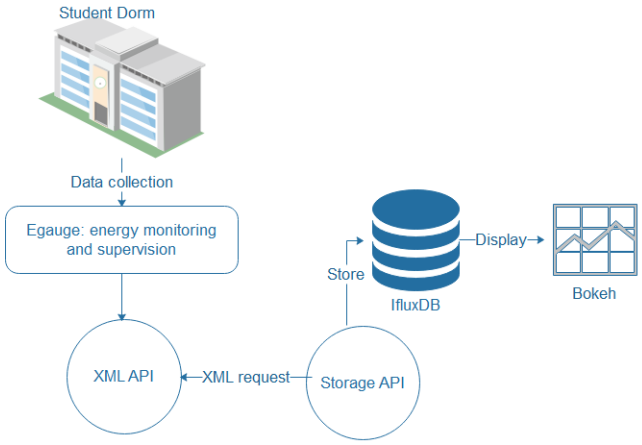


FIGURE 2.13: Data collection system actors

EGauge 47536

EGauge is an energy monitoring and supervision system. This connected sensor can measure many values: voltage, current, frequency... and calculate the powers. The box directly recovers the voltage values. EGauge has 15 ports that can each accommodate a current transformer. Ammeters were therefore connected to each port to measure the alternating current of a phase. Since the residence is wired in three-phase, the powers of 5 "classes" could be recovered: ground floor, floors 1-3, floors 4-6, boiler room, and overall incoming power. An Ethernet cable connection then connects the EGauge box to the network locally and configures it. Through the graphical interface, the sensor and registers are configured. The records are the values retrieved by EGauge and will be stored in its internal memory.

Egauge makes a measurement every second that will then be stored in its internal memory. A granularity system is defined by default: the device stores one measure per second for the last hour then performs a one-point compression per minute for the previous year and one point per 15min for the last years. The egauge offers an XML API that allows us to retrieve registered data in XML format. We can request information from the EGauge by indicating a query after the URL corresponding to the eGauge. There are two types of possible requests:

- Instantaneous query
- Stored data query

Collecting data every second generates a considerable amount of data that cannot be stored in a conventional database. The storage API retrieves every 5 minutes the values of registers in XML tree form and stores the information in an InfluxDB database. The storage API runs continuously on a server to avoid noise or blank data from appearing in the stored data.

InfluxDB [63] is a non-SQL database specializing in storing time series. It can handle a lot of data without too much writing and reading time. InfluxDB has a nano-second accurate timestamp by default and makes it easy to make time queries.

Bokeh [64] is a Python library for creating interactive visualizations for modern web browsers. It helps build beautiful graphics, ranging from simple plots to complex dashboards with streaming datasets. Finally, Bokeh is the graphical interface for displaying the data stored in influxDB. All the collected data from the student dorm will be based on this architecture.

2.5.3 Micro Grid

Classification in energy consumption is essential since we have several kinds of buildings (residential, businesses, industries...). We will focus on three classes of energy consumption: critical, programmable, and comfort. They correspond to different electric device types that may or may not be programmed during time intervals. A microgrid is a three-tier energy implant with buildings consuming energy, renewable facilities generating power, and storage means. The objective of a microgrid is to achieve optimization by matching at best energy consumption with energy harvesting. The storage plays a buffer role in network optimization.

Data set description

We possess for some data (consumption of the incubator and production of PV panels) more than three years of stored data and more than one year and a half of consumption data for the student dorm. We are considering actual consumption data collected from buildings (an incubator and a student dorm) and exact production data collected from photovoltaic panels. Various information concerning consumption are available:

- Date and time (in UTC)
- Heating
- Cooling
- Plugs usage

Likewise, for production, various information are available:

- Date and time (in UTC)
- Global_Solar_Flux
- Measured Power at Maximum PowerPoint

Fig. 2.14 shows an example of one-day energy consumption and prediction.

The chosen granularity for our experimentation is one hour.

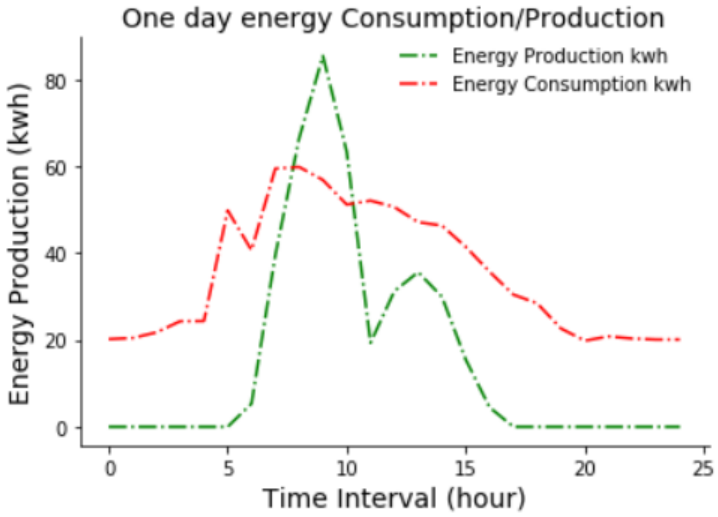


FIGURE 2.14: One day energy consumption/production

Energy classification

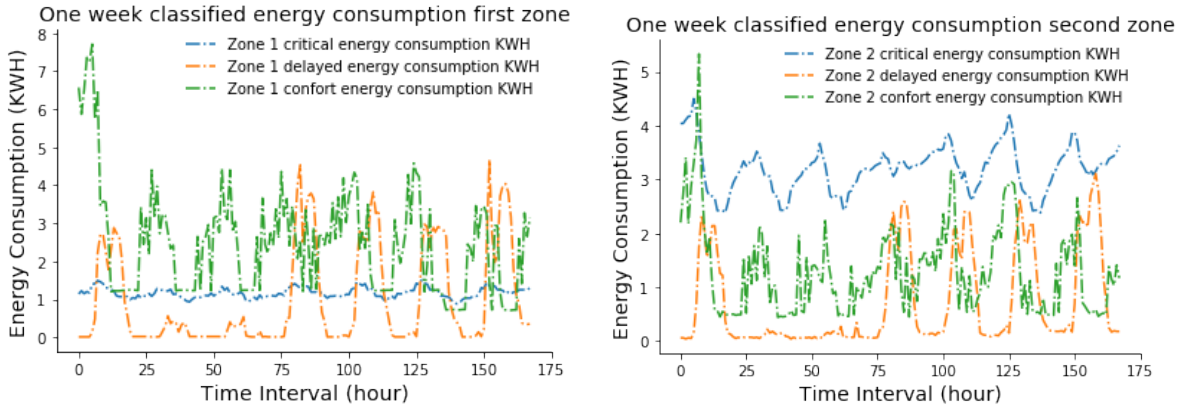
In our work, we used the classification method to distinguish between different energy production types and the kind of buildings.

We extensively used the MLP to classify energy consumption since we categorized them into three distinct categories. Critical energy consumption (CE) is the first category, which corresponds to the vital user demand. This kind of demand needs to be served without delay (in analogy with real-time applications in the data network QoS). Here, in our dataset, this corresponds to appliances such as ventilation and data centers.

The second category is the delayed energy (DE) consumption demand (like variable bit rate in computer networks) that could be served later (in the dataset, it corresponds to heating and air conditioning...). This kind of demand could be delayed in time.

The last energy category is the demand for comfort energy (BE). It corresponds to the best effort in computer networks. This demand could be served anytime during the day, on the condition you have the necessary resources. Otherwise, this type of demand may not be assured.

The second utilization distinguishes between the startup incubator and the student dorm building's consumption. In fact, due to their difference in use. One will mainly consume during the day, while for the second one, the consumption peak will mainly occur at night.



(A) First Zone (B) Second Zone

FIGURE 2.15: One week energy consumption with classification for two zones

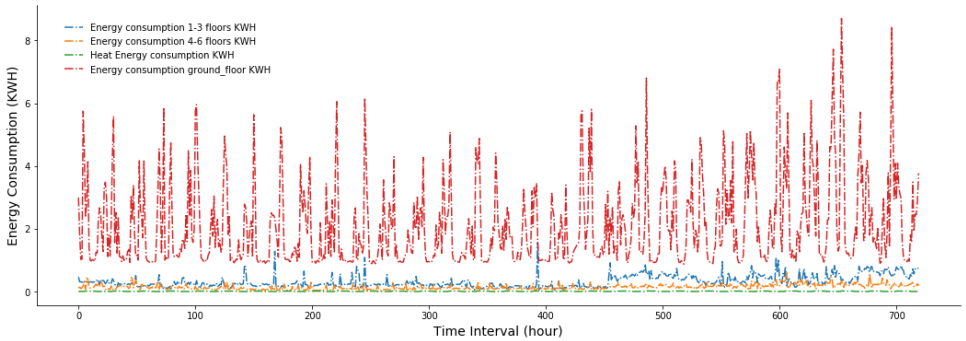


FIGURE 2.16: One month energy consumption student dorm, for the 1-6 floors, the ground floor and the heating

Energy Prediction

We studied the effects of univariate and multivariate rédiction on energy consumption. We also examined the impact of varying the shift step on prediction accuracy. Concerning the MicroGrid application, we made various predictions. The various consumption categories are studied as the total consumption of both buildings. The first studied predictions are the energy consumption demand. The second one is the energy produced with photovoltaic panels.

Different granularities have been investigated. We considered time series with one-hour granularity and time series with 10 minutes granularity. This means that we have an observation every 10 min or every hour.

The difference is when we need to make a one-day-ahead prediction with the first granularity, we will only need to predict 24 observations, while in the second case, we will need to predict 144 observations. Although it seems trivial, it affects the time and size of the NN performance.

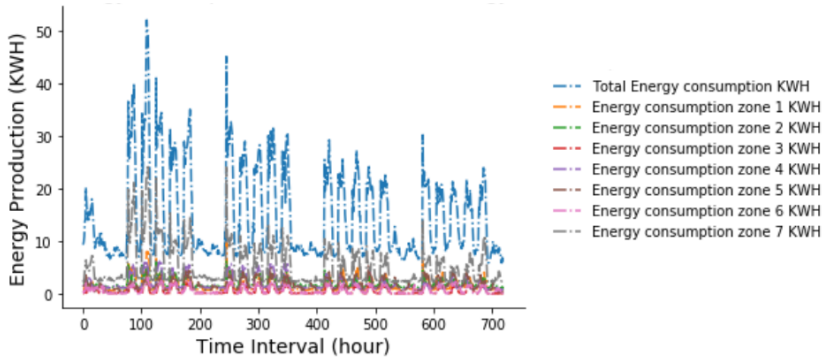
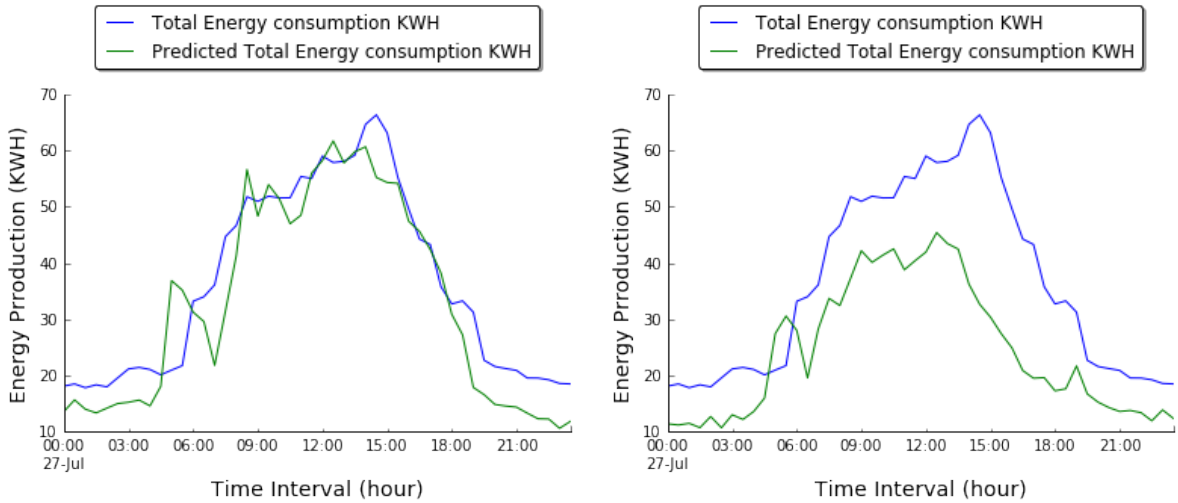


FIGURE 2.17: One month energy consumption seven zones/total energy consumption

Results

In this section, we will expose some energy prediction results.

We use the energy consumption datasets with 30 minutes and one-hour granularity to train our neural network to predict the next 30 minutes, 1 hour, 6 hours, 24 hours, and five days' energy consumption. Fig 2.18 presents the prediction results using 30 minutes of granularity as input. In the next Fig. 2.18a, the blue curve represents the actual data, while the green represents the next 30 minutes of energy consumption. The calculated MSE is equal to 0.005, and the RMSE is equal to 0.067. In the next Fig. 2.18b, we are predicting



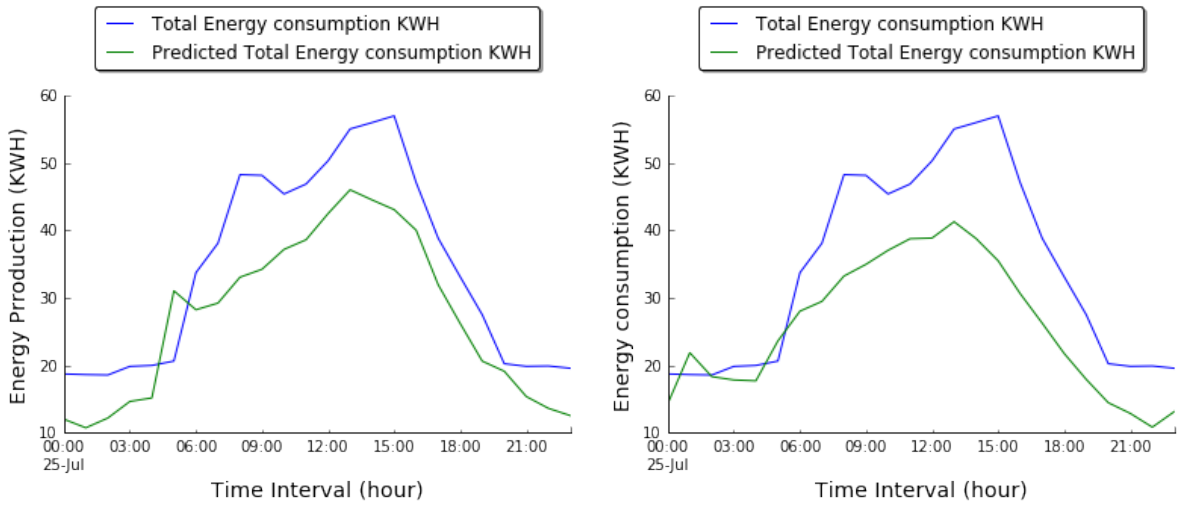
(A) 30 Minutes energy consumption real vs predicted with LSTM

(B) 24 Hour energy consumption real vs predicted with LSTM

FIGURE 2.18: energy consumption prediction with 30 min granularity input data

the total energy consumption. The blue curve describes the actual data, and the green represents the predicted energy consumption. The calculated MSE is 0.010, and the RMSE is equal to 0.100.

Fig. 2.19 and Fig 2.20 present the results of predictions using one-hour granularity as input for the training phase of the neural network. We are predicting one-hour energy consumption in the next Fig. 2.19a. The blue curve describes the real data, and the green represents the predicted energy consumption. The calculated MSE is equal to 0.009, and the RMSE is equal to 0.095



(A) One Hour energy consumption real vs predicted with LSTM (B) 6 Hour energy consumption real vs predicted with LSTM

FIGURE 2.19: energy consumption prediction with one hour granularity input data

In the next Fig. 2.19b, we are predicting 6-hour energy consumption. The blue curve describes the real data, and the green represents the predicted energy consumption. The calculated MSE is 0.009, and the RMSE is equal to 0.097.

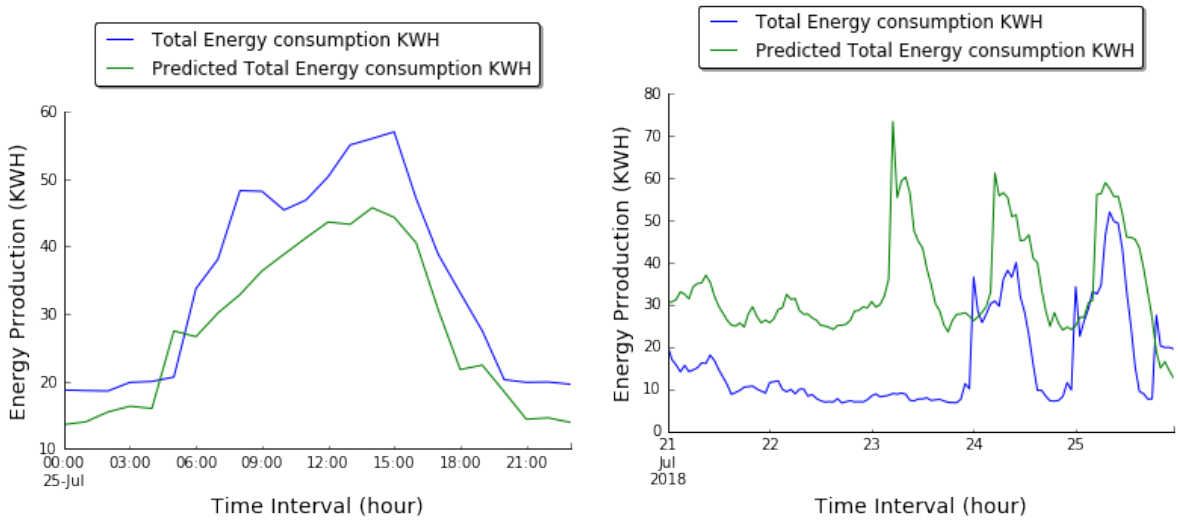
We predict one-day energy consumption in the next Fig. 2.20a. The blue curve describes the real data, and the green represents the predicted energy consumption. The calculated MSE is 0.011, and the RMSE is equal to 0.103.

We predict five days of energy consumption in the next Fig. 2.20b. The blue curve describes the real data, and the green represents the expected energy consumption. The calculated MSE is 0.014, and the RMSE is equal to 0.117.

The more the prediction will be over time, the worse the KPI will be. However, compared to the state of art, these performance measures remain, to our knowledge, very good.

In Fig. 2.21, we are predicting the energy produced with photo voltaic panels for the first zone a day ahead. The different values for MSE, RMSE, MAE, and R2 are 0.019, 0.137, 0.083, and 0.717. The blue curve describes the real data, and the orange represents the predicted energy production.

Fig. 2.22 presents the results of predictions using one-hour granularity as input for the training phase of the neural network. We are predicting one-hour critical energy consumption in the next Fig. 2.22a. The blue curve



(A) 24 Hour energy consumption real vs predicted with LSTM (B) 5 Days energy consumption real vs predicted with LSTM

FIGURE 2.20: energy consumption prediction with one hour granularity input data

describes the real data, and the green represents the predicted energy consumption. The calculated MSE is equal to 0.004, and the RMSE is equal to 0.062.

The Fig. 2.22b, we are predicting one day's critical energy consumption. The calculated MSE is equal to 0.004, and the RMSE is equal to 0.065.

Fig. 2.23 presents the results of predictions using one-hour granularity as input for the training phase of the neural network.

We are predicting one-hour delayable energy consumption in the next Fig. 2.23a. The blue curve describes the real data, and the green represents the predicted energy consumption. The calculated MSE equals 0.027, and the RMSE equals 0.164.

The Fig. 2.23b, we are predicting one day of delayable energy consumption. The calculated MSE is equal to 0.039, and the RMSE is equal to 0.197.

Fig. 2.24 presents the results of predictions using one-hour granularity as input for the training phase of the neural network.

We predict one-hour comfort energy consumption in the next Fig. 2.24a. The calculated MSE equals 0.002, and the RMSE equals 0.048. The blue curve describes the real data, and the green represents the predicted energy consumption.

We are predicting one-day comfort energy consumption in Fig. 2.24b. The calculated MSE is equal to 0.039, and the RMSE is equal to 0.197.

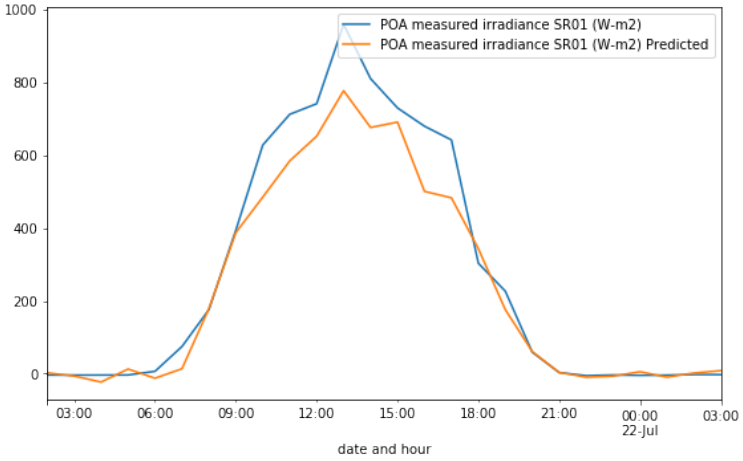
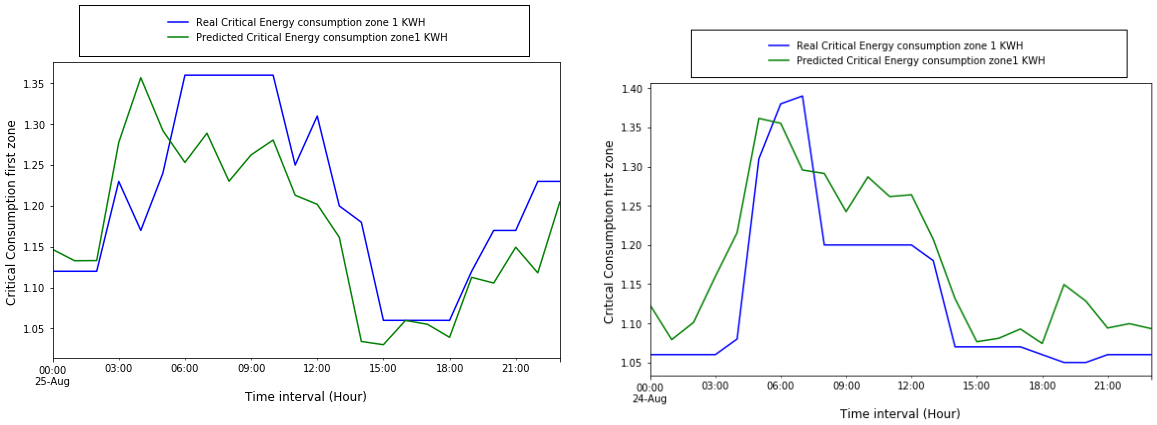


FIGURE 2.21: One day energy consumption/production



(A) One Hour critical energy consumption real vs predicted with LSTM

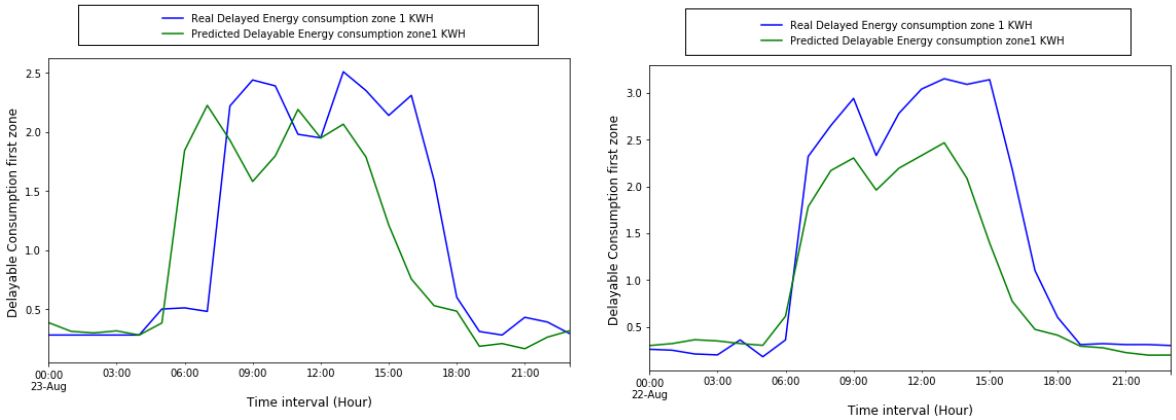
(B) One day ahead Critical energy consumption real vs predicted with LSTM

FIGURE 2.22: Critical energy consumption prediction with one hour granularity input data

2.5.4 Cellular networks

We started by extensively treating Call Detail Records (CDR) time series. They represent mobile cellular connections concerning time and cell Ids. Hammami et al.[65] succeeded in classifying them into three categories of comportment depending on the district. Dany [66] also did classification-based following features. The work was further extended to detect anomalies on such cellular networks.

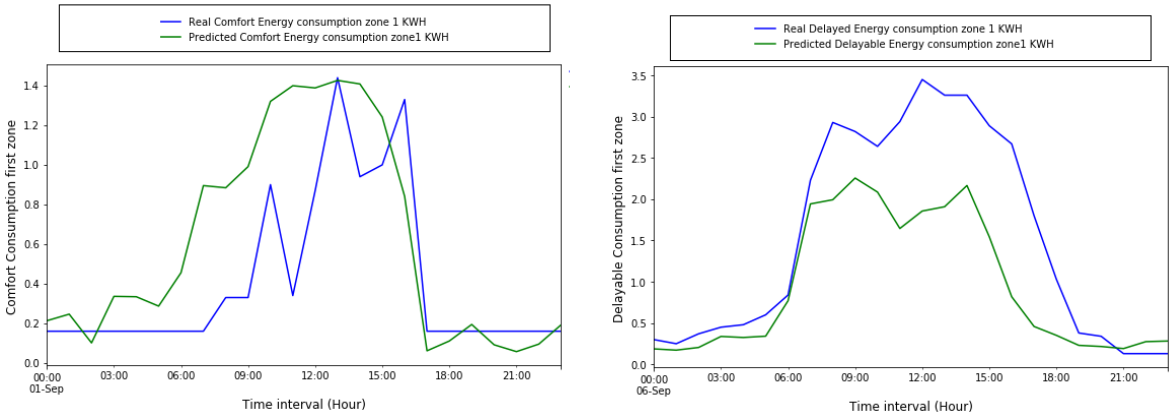
An essential requirement of those CDR traces is the capacity to detect future traffic. But we can't use the same prediction algorithms to detect traffic for all cells since it depends on the district comportment, as explained before, for building energy classification. We first have to classify so that the neural network can later make the best predictions with differently trained weights. We can also classify the traffic category per cell Id and apply the same reasoning.



(A) One Hour critical energy consumption real vs predicted with LSTM

(B) One day ahead Critical energy consumption real vs predicted with LSTM

FIGURE 2.23: Critical energy consumption prediction with one hour granularity input data



(A) One Hour critical energy consumption real vs predicted with LSTM

(B) One day ahead Critical energy consumption real vs predicted with LSTM

FIGURE 2.24: Critical energy consumption prediction with one hour granularity input data

Dataset Description

In our work, we used two distinct CDR datasets. The first one is the call detail records provided by *Telecom Italia* as part of the big data challenge context [67]. It is a rich and open multi-source aggregation of telecommunications, weather, news, social networks, and electricity data.

This dataset contains over 319 million user-activity records for 10,000 squares having a $235m \times 235m$ size spread across Milan, Italy. The records are two months from November 1st, 2013, to January 1st, 2014, and divided into 10-minute timestamps. Personal users' data are removed to preserve privacy.

Hence, the raw data contains five user-specific activity features: incoming and outgoing SMS, incoming and outgoing calls, Internet usage, and geographical location (Cell ID).

In our study, we select three zones in Milan city, and we study people's Internet activity.

- Duomo: is the downtown (city center) and presented by cell ID= 5060
- Bocconi: is a private university located at zone having cell ID=4259
- Navigli: is a touristic district known also by night life and its cell ID=4456

The internet activity in these cells is depicted in Fig.2.25 and Fig. 2.26. It is clear that in the last weekend of November 2013 (24^{th}-25^{th}), the traffic load decreased at the university and increased in Duomo square, one of the most attractive touristic places in Milan. Furthermore, end-of-year festivities present some anomalous patterns that occurred during the last week of the year (Christmas celebrations; 24^{th}-25^{th} December (Fig. 2.b)) in all cells. Another annotated anomalous traffic activity is detected at the university from 20^{th} Dec. 2013 correlates with the end of the year vacations.

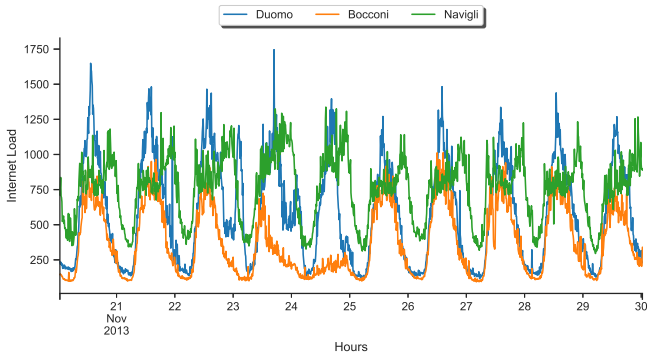


FIGURE 2.25: Internet load activity from 20^{th} to 30^{th} Nov. in the three cells

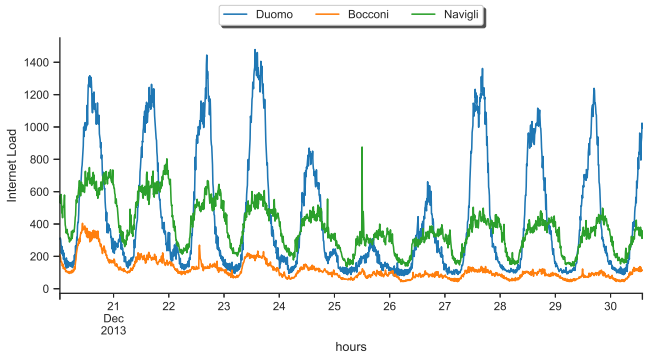


FIGURE 2.26: Internet load activity from 20^{th} to 30^{th} Dec. in the three cells

The second data set is extracted from the *D4D-Senegal challenge* project [68]. This dataset collects call detail records of phone calls and SMSs of about 9 million users during 2013.

It is divided into three components: the first set contains the antenna-to-antenna traffic for 1,666 antennas hourly. The second set includes one year of coarse-grained mobility data at district levels. The third set contains fine-grained mobility data on a rolling 2-week basis for a year with bandicoot behavioral indicators at individual levels for about 300,000 randomly sampled users.

We extracted from the last set the information allowing us to monitor the instant number of users in each cell and, thus, estimate the instant BS load (or users' occupancy) for the whole day. Unlike the previous dataset, D4D CDRs set provides user communication information at a fine-grained space scale, i.e., at each base station instead of geographic square aggregation. Hence, we can monitor each cell apart and detect network outliers precisely.

We extracted from these CDRs the times-series data corresponding to Dakar city, which describes the daily evolution of user occupancy within a BS.[69]

- Friday noon use-case: is an abnormal decrease of consumption during some hours. The time_series dataset used to validate our framework with this testbed contains almost 17,000 data points, of which 5,760 are anomalous.

- Tuesday the 5th of February use-case: is a sudden decrease in user consumption at night due to a technical problem. The data set used for validation contains almost 43,000 data points, of which 720 are anomalous.

Base Station Profile Classification

We conducted extensive studies on the classification of cellular communication and energy patterns. The methods used were based on convolutional networks, multi-layer perceptrons, support vector machines, and other advanced statistical tools. The results point out that these algorithms give quite close classifications. The choice of the best algorithm depends on several parameters, including the amount of available data to train the algorithm and the duration of training.

The internet activity in these cells is depicted in Fig.2.25. These three cells represent three different classes of behaviors and will be used for classification.

We can see from the figure that each cell has a repeated different behavior than the other two. The first cell experiences heavy traffic with peaks in the afternoon corresponding to business hours and users commuting into and from the city center. Activity is relatively stable all week.

The second cell has a very characteristic pattern with quasi-shutdown during weekends. This is typical behavior for academic locations. The third also has its particular pattern. It is characterized by a very late activity that almost catches the rise of the first class.

Those behaviors depend on many factors and differ, in our case, on the location of the time series collection. We deal here with cellular communications. It was proved that the behavior of data/call consumption presents two important facts:

- the location of the cell gives a good idea of the expected consumption pattern
- the patterns are different and should be treated distinctly when we apply prediction and regression algorithms.

From these observations stems the necessity to classify cells. Based on the previous section presenting the dataset, it comes out from our case study that three classes would cover most of the behaviors in a sizeable capital-grade city such as Milan. The classes group similar behaviors, but the intensity of the connections is not a parameter of choice. All cells are normalized to focus only on the pattern shape.

So we adjusted the analysis on the three classes as explained before. To know why we can stop at three categories or define more, statistical and subjective methods were developed in [70].

We use a Convolutional Neural Network of One Dimension (CNN-1D) to classify the cells in the city. This network comprises two convolutions, two pooling layers, and one dense flattening layer to output three classes in the form of a Softmax probability. Here, we restrict our study to one input feature: the internet traffic for individual cells. Classification is important for operators and network providers as it eases planning, the study of key performance indicators, and targeted marketing. Classification and transfer learning provide an excellent couple to enhance cellular network management.

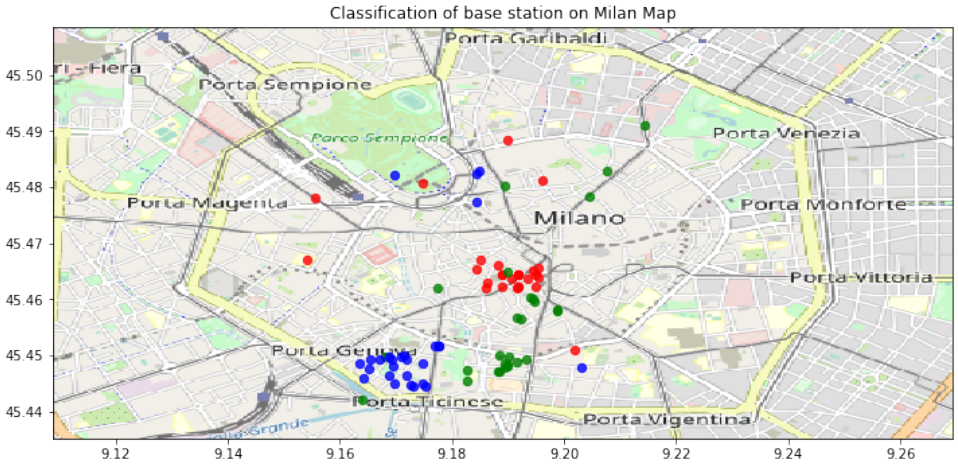


FIGURE 2.27: Classification results visualization over Milan Map

Figure 2.27 shows the results for classifying 81 cells according to the previously defined three classes [71]. The red spots present the downtown city

class, which corresponds to Duomo. The green positions represent the universities according to the used dataset Bocconi. And the blue points correspond to tourist districts, also known as nightlife districts, represented by Navigli. As mentioned, a deep analysis of classification patterns could be found in [66].

Base Station Load Prediction

Concerning the cellular application, we made various predictions. The first studied predictions are the load of a determined base station in Milan.

Different granularities have been investigated. We considered time series with one-hour granularity and time series with 10 minutes granularity.

We also predicted the load of the base station in Dakar city. We choose one BS. and train the neural network on the historical data.

2.5.5 Public Transportation

Prediction is not restricted to 1D data. It could be used for multi-dimensional time series. The route is an example of a 2D dataset. We try our LSTM network to this kind of data.

Dataset Description

We use six months of taxi traffic traces in Rome (Italy), where each day contains approximately 70000 different vehicle trajectories. From this data set, we analyze the trip lengths. Trips are periodically sampled by GPS, giving a good approximation of the trip trajectory. Each trip is truncated to 20 GPS points with time. That is why in our learning method, we describe trips by a 20x3 dimensional array, each position represented by the user's latitude, longitude, and time.

The goal is to use the above-described RNNs to make future predictions of the trip directions and durations. Once this information is generated from the neural network, it is easy to feed it to the vehicular models that will evaluate the necessary energy for traction and comfort in those driving conditions. Although weather conditions are known and can give us the required comfort energies for different driver profiles, the driving styles are missing (calculations can be made for other driver profiles).

Trajectory Prediction

From the dataset, we could analyze the trip lengths. GPS periodically samples trips, but the recorded time stamps are not periodic, probably due to some phenomena like jitter. Nevertheless, it gives a good approximation of the trip description. We call a "hop" each time a base station records a user.

For each car trajectory, we keep 20 hops that define the trip. Each jump is identified by time, latitude, and longitude.

As explained, data are presented as time series to the neural network. We build a neural network consisting of several cells. Each cell hosts several

hundreds of hidden layers. The output of the RNN is concatenated into one or several dense layers to improve the precision.

In the presented work [72], we decided to shift our data by four steps. Given the current position, the ML algorithm must predict the successive four positions of the car during the trip. The prediction contains 3 phases:

- In the training phase, if the input layer contains the ten first values of one trip, the label will include the ten values shifted by 4.
- In the test step, the ML algorithm will predict the four following values of the trip based on the input layer. If the shift step increases, the algorithm's performance will decrease or need more training.
- The last step consists of normalizing the data because NNs work better on values between -1 and 1.

Since the data is not a typical periodic waveform but rather a route shape, we highlight the graphical GPS representation of predicted routes compared to real ones for several algorithms. In the next section, we present the results of predicting four trajectories using Dense Long Short Term Memory and Dense Gated Recurrent Units.

Results

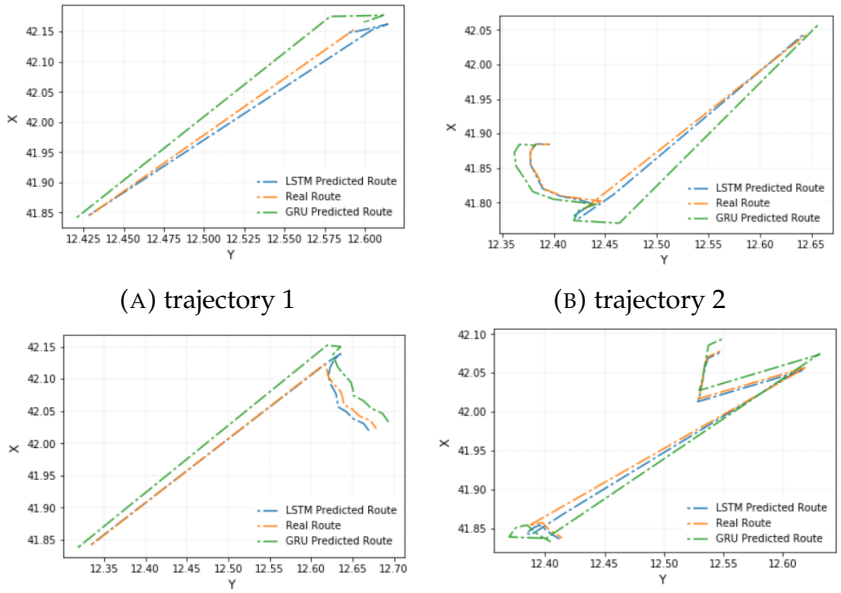
Table 2.5 gives the results for several taxi routes in Rome (from 1 to 4) and compares LSTM to GRU.

	GRU		**LSTM**	
Route	MSE	RMSE	MSE	RMSE
1	0.00011	0.01056	0.00002	0.00435
2	0.00011	0.01029	0.00006	0.00797
3	0.00015	0.01242	0.00003	0.00504
4	0.00009	0.00973	0.00008	0.00871

TABLE 2.5: Obtained measures for each trajectory for mean square error (MSE) and root mean square error (RMSE) in the test phase.

The results obtained in table 2.5 are comparable. Among the four trajectories, LSTM has less MSE and RMSE than GRU. Hence, LSTM gives better results in route predictions. For example, for trajectory 1, the MSE and RMSE of GRU are 0.00011 and 0.01056, respectively. However, the MSE and the RMSE of LSTM are equal to 0.00002 and 0.00435, respectively.

Figure 2.28 presents the obtained route predictions with the application of Dense LTSM (Blue curves) and Dense GRU (Green curves) and the actual taxi route in the test phase (Orange curve). It is clear that LSTM is very close to real traces in all the routes, and GRU stays farther to real traces in all the courses and especially in trajectory 1 (Figure 2.28a) and trajectory 2 (Figure



(A) trajectory 1

(B) trajectory 2

(C) trajectory 3

(D) trajectory 4

FIGURE 2.28: Forecasted trajectories using DLSTM & DGRU

2.28b) and trajectory 3 (Figure 2.28c). We confirm these results with the values of RMSE obtained for these trajectories. On the other side, we notice a smaller LSTM RMSE value for 2.28a and in 2.28c. It is equal to 0.01056 and 0.00504, respectively.

In Figure 2.28d, it is clear that LSTM outperforms GRU, but both curves are close to real traces. This is because the values of RMSE and MSE are also very close. We notice the MSE of LSTM is 0.00009, and GRU is 0.00008. In addition, the RMSE is equal to 0.00973 for GRU and 0.00871 For LSTM. We conclude that LSTM gives better results than GRU. The predictions, in general, are very pertinent compared to the real information.

2.6 Discussion

We evaluated our proposed methodology in three different application fields. First, we evaluated various meta-parameters to choose the more accurate one. After that, we applied our prediction algorithm and evaluated the results based on multiple error metrics.

2.7 Conclusion

We presented in this chapter our proposed architecture For Time Series Analysis. Since we want to propose a general methodology for time series, we applied our method to three different time series types.

We started by studying the use of classification to categorize the time series. We began by identifying the most used one. Then we compared their

performances according to our data sets. This classification allows us to better choose the recurrent neural network for predictions.

After that, we studied time series forecasting. We compared using three algorithms: SVR, GRU, and LSTM. Our experimentation showed that LSTM performs better than the two others. GRU is, however, faster in the training process.

Then we described how we collect one kind of time series. Sensors were installed in a student dorm building, allowing us to expand our database of electricity consumption.

Finally, we showed some results for each studied time series data.

To our knowledge, these published contributions were among the first to apply NN to time series.

Chapter 3

A Deeper Time-series Analysis of the selected applications

3.1 Introduction

In this chapter, we exploit a more advanced feature that can be found in RNN, and we propose novel applications that can benefit from RNN characteristics. The first application is semantic compression, where we emphasize the need to compress and only transmit relevant information using a particular network: LORA. The second application is transfer learning. We will show the efficiency of this method while dealing with the lack of data and data personalizing. The last application described in this chapter is anomaly detection.

3.2 Semantic compression

Data compression is now a very mature subject. Globally, we find data compression for storage reduction and transmission in general. The venue of modern video transmission introduced lossy compression based on quantization. Here, the idea is to adjust transmitted values to fixed thresholds to reduce compression states. One can refer to [73] for details.

Prediction can be considered as a compression means. The system, regarded as a black box, works as follows: When an input is entered, the next k values are output. Several modern regression techniques exist (mainly support vector families and neural networks). We focus on recurrent neural networks for their high performance in prediction.

3.2.1 Introduction

LORA networks represent a good candidate for IoT data transfer. They scale well, provide new naming techniques, and do not consume large amounts of energy. Yet, they need to offer sufficient bandwidth for a large panel of IoT devices, especially regular time-based sensing devices such as power consumption, temperature, pollution sensors, etc.

Our experience with a medium-scale power sensing deployment in administrative premises has shown that the main issue in this deployment was the communication part. The link between many sensing devices and the

backend has been a real issue. Although it was easy to lay down sensors and small embedded PCs, deploying a specific LAN or a WiFi network for such a purpose was difficult (both technically and administratively). Cellular connections can be a good solution, but they would cost too much. So our left choice was to use a cheap Long Range (LORA) connection.

LORA has a significant bottleneck in its limited bandwidth. LORA groups in Internet Engineering Task Force (IETF) work intensively to provide compression mechanisms on the protocol to make it more efficient. This work is great for our problem but needs to solve the dilemma of sending megabytes of data daily.

Data compression is now a very mature subject. Globally, we find data compression for storage reduction and transmission in general. The venue of modern video transmission (MPEG) introduced lossy compression based on quantization. Here, the idea is to adjust transmitted values to fixed thresholds to reduce compression states. One can refer to [73] for details.

Prediction can be considered as a lossy compression means. The system, considered a black box, works as follows: Several modern regression techniques exist (mainly support vector families and neural networks). We focus on recurrent neural networks for their high performance in prediction. When an input is entered, the following k values are output.

3.2.2 Related work

We investigate using the neural networks prediction technique as a substitute for classical compression. The idea is a corollary of the successful usage of recurrent neural networks in prediction. Since these tools provide an excellent way to make time-series predictions, we could use the same configured network to replace data transmission. So instead of sending raw data from the IoT sensor, we train a neural network in the IoT neighborhood and only send its weights. On the other side, when an abnormal situation is detected in the generated data, it is directly sent as an outlier. This will not constitute an overhead in bandwidth as it is usually a rare event.

In [74], authors propose adding machine learning techniques at the edge device to perform low-power transmission through LORA. They propose sending only the final output via LORA.

In [75], authors propose a method to determine offloading and transmission strategies that are better for directly sending fragmented packets of raw data or sending the extracted feature vector or the final output of deep learning networks, considering different operational performance metrics.

In another work, [53], the authors aim to study LORAWAN traffic compression, precisely its data payload, using a machine learning-based compression scheme. A deep LSTM algorithm was developed and integrated into a small, constrained hardware system that allows efficient compression while keeping the user within a reasonable error margin. The authors built an experimental test bed to check the capabilities of the onboarding LSTM algorithm on-sensor to forecast data, achieve dual prediction, and eventually compress data traffic and save energy.

Classical compression approaches based on dictionaries or entropy coding have been adapted to IoT, like [76], where a specific dictionary is created for different data depending on their change frequency.

When the IPv6 protocol conveys data, the draft gives pretty good compression results. LORA research groups in IETF [77] have recently proposed a header compression mechanism for LORA. We are involved in this work as well, and we consider that header compression for LORA, in general, will be beneficial to all kinds of transmission, whether using our semantic compression technique or not.

3.2.3 Proposed method

IoT data collection requires a large bandwidth for its transmission. We propose hence to divide the data collection process into two parts.

In Fig. 4.3, we show the main stages of the new architecture. First, we go through data collection from the sensor network. Then, the dataset is used locally in the gateway for the training [78].

The resulting prediction is then compared to the newly collected data. If there is a significant difference, the collected data varies enormously compared to the predefined threshold the system will detect an outlier. The value will be sent over the LORA network to permit the reconstitution of the dataset with the anomalies.

Training output is a set of weights described above and a periodic sample of the raw IoT data. Combining the raw data as a seed and the neural network will mimic the time series with a tiny error (hence the semantic name compression). They can be re-trained frequently if the time series change.

The figure shows data coming from sensors and inputs. The inputs correspond to the context information used for prediction. Here, we have the day of the week, minute of day, season, inside temperature, and outside temperature. The inputs are as important as the data.

Outlier Detection and IoT over LORA transmission

The prediction helps in the outlier detection process. Outliers help detect failures and recommend parameter tuning to avoid the malfunctioning of the IoT. The idea is that we have a variable threshold making an envelope on the predicted signal. When the real value measured at the sensor overpasses this threshold, it is considered an outlier and sent directly over LORA as an alarm.

Transmission and data restitution

The IoT over LORA transmission part is straightforward [79]. A burst containing the exact (i.e., optimal) neural network weights is sent periodically with a timed sample of the raw data.

We can additionally use the LORA header compression draft for more performance. In the real deployment of the proposed model, outliers or

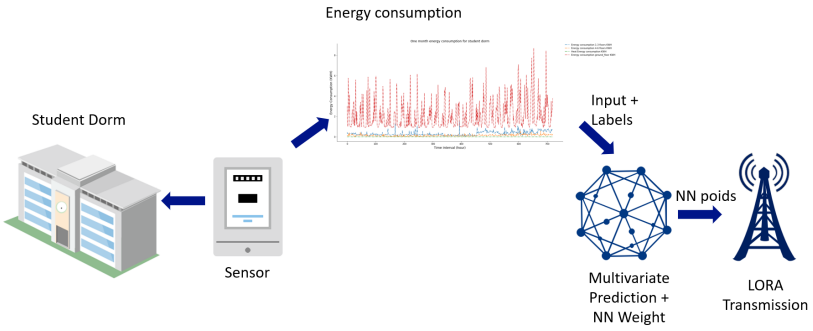


FIGURE 3.1: The different steps of the proposed prediction approach

anomalies are detected separately and sent with a different tag for immediate treatment. Moreover, depending on the capacity of the LORA messages, the maximum amount of information about the time series is sent.

The received weights are reloaded into a python program consisting of the same neural network as the training one. The output is generated based on inputs (typically the day of the week, temperature, and time of the day) and very rare data samples. The output, in our case, is a predicted consumed power for the input parameters presented to the test neural network. It is considered a semantic compression since the difference between the generated data at the sensor and the one restituted is equal to the training error. Data is rarely sent over the network, mainly its neural network representation.

In [53], A deep LSTM algorithm was developed in collaborative work and integrated into small, constrained hardware. The result shows that training a neural network locally can efficiently minimize the traffic while preventing non-relevant transmissions with a significant impact on energy consumption. We observed the effect of the neural network size and the decision threshold on the compression ratio and the MAPE. The system allows efficient compression while keeping the user within a reasonable error margin. It can be customized depending on precision and compression trade-off requirements. This work is continuing today to test smaller IoT.

3.2.4 Performance analysis

This part analyzes the proposed methodology's performance through its steps. As mentioned, RNN networks are not used frequently and have been replaced by other methods that perform better. We use basic RNN, LSTM, and GRU as compression candidates. We noticed that GRU has smaller weights as it presents fewer gates for the same configuration as an LSTM network. However, LSTM gives much more precise prediction and hence better data restitution.

The neural network weight size for each prediction technique is presented in Table 3.1, where we observe that the GRU produces notably smaller sizes. Recall that the edge gateway sends only the compressed (e.g., using the zip

Prediction model	*NN weight size*	*MSE*
Simple RNN	19.8 kB	0.0403
LSTM	30.6 kB	0.0406
GRU	27 KB	0.0714

TABLE 3.1: Comparison (10 epochs, 3 steps per epoch and 30 neurons) of RNN, LSTM and GRU neural network sizes

function) neural network weights. The table shows that in this kind of time series, RNN has a small Mean Square Error compared to LSTM and GRU. GRU is the best in compression size and training time but with a higher restitution error.

Moreover, in Tab. 3.2, we show different results of the neural network size before and after the online training process. The experiments show that before the training step, the most influencing meta parameter on the neural network size is the number of neural network layers. After the training step, the neural network size will increase according to the number of iterations per epoch. While testing the various neural network configurations trained with several meta parameters, we noticed that the configuration (conf16) gave the best results. According to the proposed key performance indicators, Mean Squared Error (MSE), Root MSE (RMSE), and Mean Absolute Error (MAE).

TABLE 3.2: Neural network sizes with different configurations

Id	*Nb of neurons*	*Layer*	*iter*	*Size-B-Training (KB)*	*Size-A-Training (KB)*	*MSE*	*RMSE*	*MAE*
1	1	1	10	194	197	0.894	0.946	0.921
2	1	1	100	194	222	0.236	0.486	0.478
3	1	2	100	327	354	0.154	0.392	0.339
4	5	2	100	327	354	0.066	0.256	0.203
5	5	5	100	725	753	0.073	0.27	0.214
6	10	5	100	725	753	0.059	0.244	0.186
7	100	5	100	731	759	0.017	0.13	0.086
8	100	10	100	1401	1429	0.021	0.146	0.101
9	100	10	1000	1401	1677	0.005	0.07	0.053
10	1000	10	500	1419	1557	0.004	0.067	0.052
11	1000	10	4000	1419	2539	0.003	0.055	0.042
12	100	100	100	13579	13606	0.02	0.143	0.099
13	100	100	1000	13579	13855	0.009	0.093	0.068
14	500	100	100	13755	13783	0.01	0.101	0.075
15	100	10	4000	1401	2521	0.004	0.062	0.044
16	100	10	6000	1401	3084	0.003	0.053	0.039

LORA bottleneck

Finally, LORA bandwidth remains the most critical parameter in the compression process. LORA will limit the amount of data that can be sent periodically. Hence, it dictates raw data transmission frequency, kind, and size of neural networks. LORA bandwidth varies from one operator to the other. The more frequently we train the network, the more we will require bandwidth to send new weights. As for outliers, it is not a transmission bottleneck since an anomaly, by definition, is a rare event.

3.3 From Lack of data to Transfer Learning solution

3.3.1 Introduction

Transfer Learning (TL) [80] is a crucial method for industrializing neural networks. In short, a trained neural network is a set of interconnected specific neurons and their collection of weights.

If we want a large-scale integration of these architectures in our daily life, we need quickly adapt their usage to our specific context. However, complete training is known to be time-consuming and requires excellent artisanal skills to build and design the best architecture with its meta parameters. Transfer learning is a second-stage re-training procedure that does not require much time or skills as the network is already specified.

As an elementary example, if one buys an intelligent vacuum cleaner. It could be already trained for primary movement sequences but needs a fast re-training (or transfer) for a particular new apartment. This section focuses on time series representing key performance indicators (KPI) in modern cellular networks, particularly edge networks. We study the adequacy and advantage of using this technique to improve classification and prediction.

3.3.2 Related work

In [81], the authors propose to use transfer learning to improve prediction. They use convolutional networks for that purpose and train on several datasets. Their main contribution is introducing the Dynamic Time Wrapping technique to help predict. First, we claim that time series best deal with Recurrent Neural Networks (RNNs) rather than CNN's. Then, we propose using CNN in the classification rather than the DTW in our work. As we previously proved, DTW gives worse classification results than classical statistical tools (SVM) [82] and hence even worse compared to CNN.

In [83], authors utilize transfer learning to classify the images of magnetic Resonance Imaging (MRI) so that they could detect and classify the subject having dementia. They used a fine-tuning pre-trained convolutional network, AlexNet. The algorithm presented encouraging results by giving the

best overall accuracy of 92.85% for multi-class classification of un-segmented images.

[84] presents a study on bi-LSTM to apply transfer learning on time series. This work is closer to our proposal but does not include classification. Moreover, context and RNN are entirely different.

Authors in [85] used TL for driver behavior modeling. They combined the dynamic time warping (DTW) and local Procrustes analysis (LPA). Sufficient data for every driver are not available. DTW discovered the relation within the datasets for several drivers. And LPA transfers the data in the historical dataset to the dataset of the target driver.

In [86], a study uses small amounts of data and benefits from transfer learning techniques to detect misogynous tweets. The approach has some approximate similarities with the first part of our architecture. However, it is targeted at natural language analysis and pattern recognition, which is not in our scope of action. Also, its objective, detecting tweets, does not match our goals.

In [87], since data acquisition is a challenging problem for Vehicular Ad-Hoc Networks, the authors proposed a Q-learning technique to make the collecting operation more reactive.

In another study, authors [88] adopted the Transfer Learning procedure for the automatic detection of the Coronavirus disease. The used dataset contained X-ray images from patients with typical bacterial pneumonia, established Covid-19 disease, and ordinary incidents. The results show that the proposed approach may have notable effects on the automatic discovery of features from X-rays related to the diagnosis of Covid-19.

3.3.3 Transfer learning Architecture

This technique is getting a tremendous natural interest in the artificial intelligence domain for many reasons. First, it is the way to deploy, on a vast scale, all kinds of neural networks. The idea is to outsource neural networks that have been trained in labs but need personalizing for the new application user. So a quick tuning would be ideal, and this is precisely what transfer learning does. Second, in many cases, we have unequal sets of data. Suppose we succeed in starting training with a considerable amount of data to sculpt the main neural network parameter shapes and tune the resulting weights with a smaller set of data. In that case, this helps in many situations, especially in the telecommunication domain.

Finally, One could need an 'Anytime computing' system that can be stopped and retrained later. This is also a feature of our architecture. We will explain all these techniques by applying them to a real telecommunication dataset. In the subsequent paragraphs, we describe our transfer learning architecture for time series. We explain how transfer learning is used in two applications: Shortage of data and personalizing. Then we detail the core dataset used for our neural network architecture training and testing.

The architecture shown in Figure 3.2 is composed of three blocks—the first consists of classifying the datasets for a better prediction. Classification

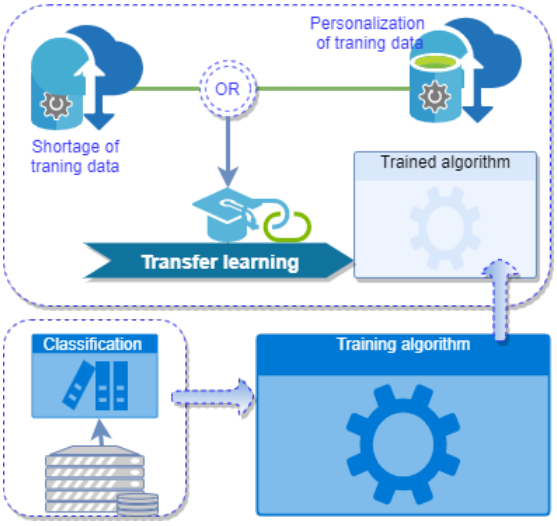


FIGURE 3.2: Transfer learning architecture

is commented on later. The second block consists of a recurrent network trained with a complete and valid dataset. We restrict the work to Deep LSTM, but other networks could be used.

The third part is proper transfer learning. It is based on the initially trained network and a new target network where knowledge will be transferred. So we have two neural networks in the architecture: an original and a target (depicted as training and trained in the figure) [71].

Shortage of training data

The first use of transfer learning in our context is necessary when the dataset corresponding to the cell where we want to make predictions is not sufficient or erroneous. A simple example of such a situation is an operator of any kind that has installed a new facility. There is a need to start the prediction process to calibrate key performance indicators, but more data must be collected. Also, one can imagine that we want to begin anomaly detection on that new facility, but we did not yet get enough data.

So we start training a neural network on an original dataset (see Fig. 4.16). It should be complete but not necessarily correspond to the same class as the targeted new facility dataset. We use Long Short Term Memory with deep layers (D-LSTM) for all the transfer learning processes.

When the training process ends with a satisfactory mean square error (or any other error metric), we start the 'transfer.' This process consists of taking the original network's trained weights and using them as a start for subsequent training that will use the small amount of available data. The final result, as it will be shown in the performance section, is sufficient to start the required procedures.

From a generic neural network to a personalized one

Personalizing neural networks is the second usage of transfer learning in the time series context. Artificial intelligence is penetrating all the fields in everyday life. Logically, we cannot expect that professionals will be capable of training every neural network in the world... So, a transfer of learning would be a desirable solution. The scenario here is to 'prepare' the neural network for behavior that we believe to be close to the target usage. Then an additional short training would help personalize the resulting system and make it fit the new usage perfectly.

3.3.4 Cellular network application

We consider the call detail records provided by Telecom Italia as part of the big data challenge contest. In this section, we assess the use of transfer learning according to the following error metrics: Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and R squared (R^2).

We will start by examining the use of transfer learning when there is a lack of data by making predictions using two granularity measures (10 minutes and 1 hour). Then, we will investigate the contribution of data specificity by dealing with transfer learning upon data from different and same classes.

We used the classification result defined in the previous section 2.5.4 where we classified 81 cells in Milan city. The result of this classification is used in the choice of cells for the TL.

Shortage of data

This series of tests demonstrates the benefits of transfer learning when the target cell has insufficient data for proper prediction training. The evaluation procedure led here is as follows:

- In this use case, we suppose there is a shortage of data for a particular cell that belongs to a specific class, which means we do not have enough data to train a neural network efficiently.
- First, we classify our data set so that we can distinguish between the different cells and to which class they belong.
- Once we identified a cell with sufficient data information that belongs to the same class as the target cell.
- We train the first neural network with data from this cell. Then we get its weight
- We create a new data set that combines the two cells' information.
- We re-train a second network after restoring the weight with two months of data composed from two different cells (same class and target cell)

TABLE 3.3: Efficiency of transfer Learning with one hour granularity

Metric \ Approach	With TL	Without TL
MSE	0.003	0.018
$RMSE$	0.053	0.134
MAE	0.040	0.104
R^2	0.774	0.598

- We train another network with one month of data derived from the target cell.
- We compare two granularities: 1 hour and 10 minutes.

We compared the predicted amount of data provided by the algorithm using transfer learning (orange curve) and the predicted amount of data provided by the second trained network (green curve) (Fig. 3.3 and Fig. 3.4). The corresponding precise error tables are: 3.3 and 3.4.

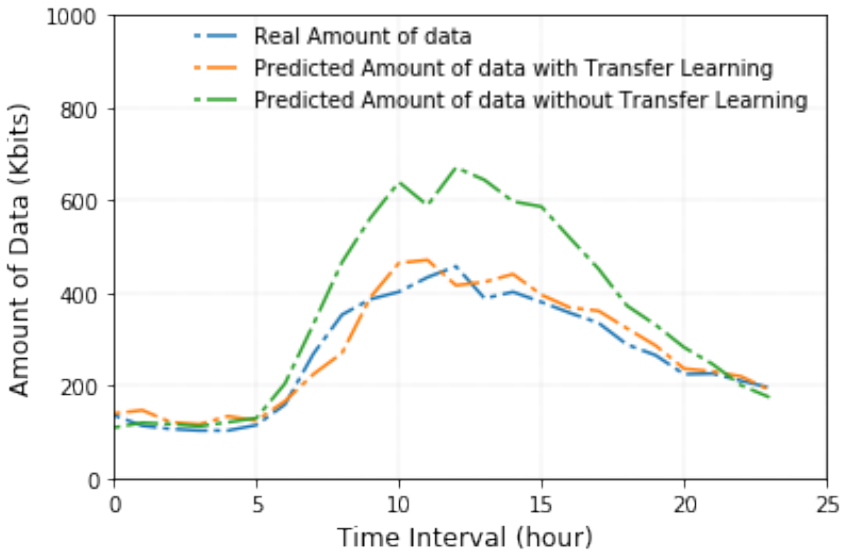


FIGURE 3.3: The pertinence of transfer learning use with One hour granularity prediction

Data personalizing

We describe hereafter transfer learning performance for two cases:

- IntraClass is when we use datasets from the same class (recall from the previous sections that a class groups cells with similar behavior) to transfer the learning

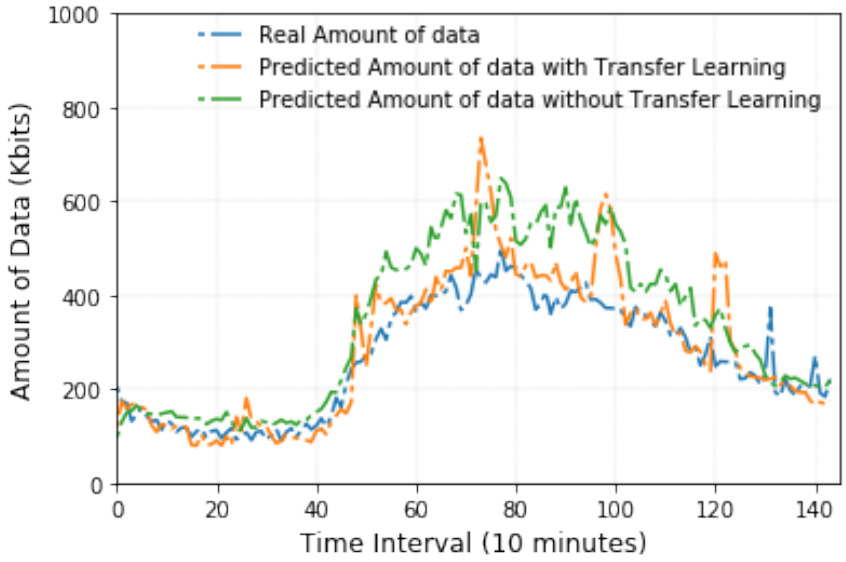


FIGURE 3.4: Improvement via transfer learning with 10 minutes granularity prediction

TABLE 3.4: Improvement via transfer Learning with 10 minutes granularity

KPI \ Approach	With TL	Without TL
MSE	0.003	0.012
$RMSE$	0.054	0.108
MAE	0.035	0.086
R^2	0.699	0.612

- InterClass is the process of transferring learning between two different classes

The adopted process is as follows. We start training the original network with a two-month dataset for a given cell in a given class. The second step consists in making the transfer learning to predict data for a new cell (we call the target cell), either IntraClass or InterClass.

We mimic the lack of training data during transfer learning to show the benefits of our transfer learning architecture on prediction for the target cell. We vary the percentage of the training data for the target cells from 10% to 40% of the total period (it consists of two months of traces).

Figure 3.5 presents the results of transfer learning using two cells that belong to the same class. In table 3.5, we compare the prediction accuracy in terms of different error metrics. The results are compared to the blue curve representing the real values. The higher the percentage of target training data, the better the results will be. The error results are summarized in table 3.5.

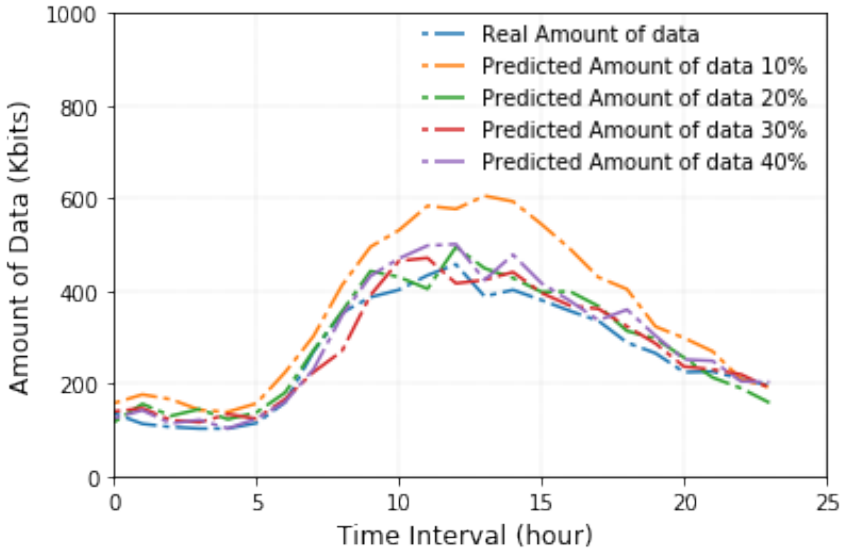


FIGURE 3.5: IntraClass Transfer Learning

TABLE 3.5: Efficiency of Intra Class Transfer Learning algorithms (university)

KPI \ Class	10%	20%	30%	40%
MSE	0.007	0.004	0.003	0.002
RMSE	0.081	0.061	0.053	0.046
MAE	0.065	0.049	0.040	0.038
R^2	0.690	0.733	0.774	0.859

Figure 3.6 presents transfer learning results using two cells belonging to different classes (here, downtown versus universities behaviors). Similarly, we vary the percentage of the used dataset during transfer learning from 10% to 40%. In table 3.6, we compare the prediction accuracy in different error metrics. We have the same conclusion as the previous test. The error results are summarized in table 3.6. We can see that our architecture improves the prediction even between different classes.

Figure 3.7 presents transfer learning results using two new cells belonging to different classes (university to nightlife). In table 3.7, we compare the prediction accuracy in terms of different error functions. Results improve proportionally with the data amount of the target cell. When the data size of the cell is small, using data from a cell that belongs to the same class will provide a better prediction.

3.3.5 Discussion

We can conclude from the above tests that transfer learning is of great utility for the two proposed applications: personalizing and shortage of data. The error tables confirm the importance of classification in the architecture. We

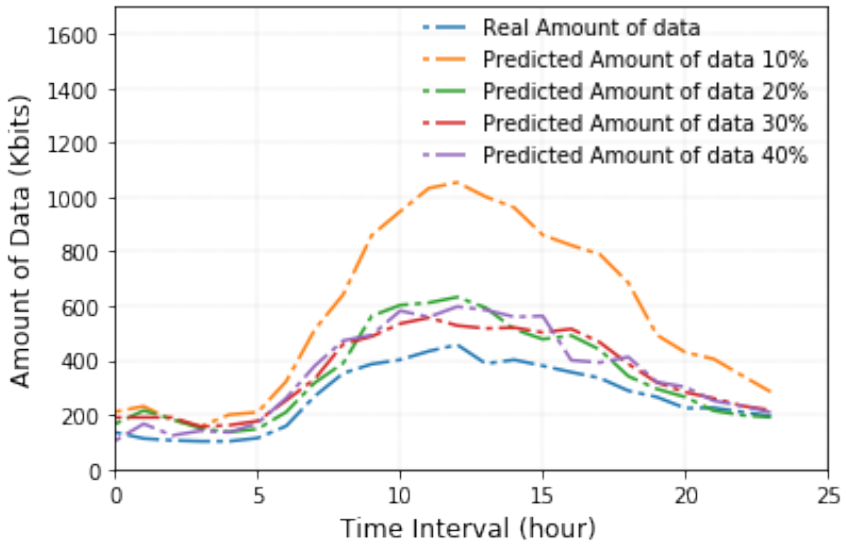


FIGURE 3.6: InterClass TL: downtown to university

TABLE 3.6: InterClass results (downtown to university)

KPI \ Class	10%	20%	30%	40%
MSE	0.073	0.004	0.004	0.002
$RMSE$	0.270	0.061	0.060	0.045
MAE	0.214	0.049	0.052	0.033
R^2	-0.629	0.594	0.550	0.747

can make transfer learning between classes to omit classification at the bottom line. But, it is still better to transfer the learning to the same class rather than to different classes.

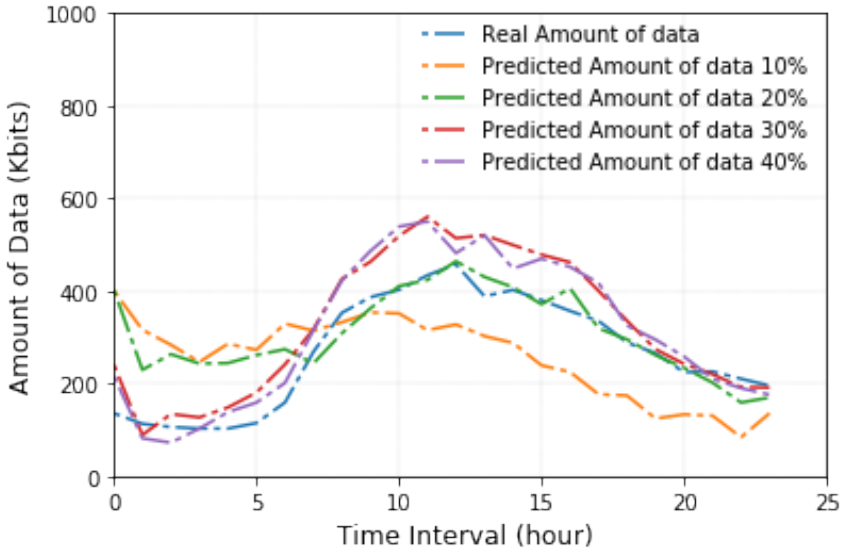


FIGURE 3.7: InterClass TL: university to night life

TABLE 3.7: InterClass TL between university and night life classes

Metric \ Class	10%	20%	30%	40%
MSE	0.011	0.004	0.003	0.002
$RMSE$	0.103	0.060	0.052	0.047
MAE	0.092	0.043	0.041	0.036
R^2	-0.751	0.431	0.752	0.826

3.4 Anomaly detection

3.4.1 Introduction

Anomaly detection, or outlier detection, in network management, has become of greater interest due to the rise in sensitive new applications in many fields, such as medicine, economics, energy, telecommunications, etc. Data mining techniques that have been recently developed can help process large volumes of datasets in a scalable and low-complexity manner compared to old classical statistical methods. Data mining combined with artificial intelligence assists in designing very efficient detection algorithms. Thus, many promising applications can be revisited, especially for detecting outliers.

In the past decade, notable general system failures have occurred in significant telecommunications operators. In July 2012, Orange Telecom, the official national French operator, suffered a severe breakdown that left 26 million subscribers unable to make calls, send or receive texts, or use data services for approximately 9 hours. The failure also affected the Orange Mobile Virtual Network Operators (MVNOs) and the interconnection with other network operators. Due to the lack of adequate management tools that can

rapidly detect this kind of anomaly, the operator could not avoid the blackout in its network or even mitigate its impact. As a result, they were forced to deploy more resources to fix the failure, adding extra costs to their subscribers' dissatisfaction. Investigations showed that an earlier update of a software stack was the origin of this blackout. The anomaly had not been notified by any alarm signal (it was probably identified as a true positive alarm).

From this incident and many others, it is recommended to upgrade the management and alarm systems with efficient automated techniques that, by analyzing real-time traces, can detect on-the-fly network anomalies. These tools can also help the operators to monitor their infrastructures and manage their networks more accurately. Strengthened by their learning capabilities, they avoid the lengthy and fastidious hand work to build evolving traffic profiles.

Network outlier detection techniques aim to automatically identify and detect abnormal and anomalous patterns which differ from the expected behavior or may present a local deviation from the standard data. In our contribution, we address the problem of detecting outliers within radio access networks.

The exponential growth of mobile devices and mobile phone subscribers has massively increased data generation, which in turn causes the creation of several spatiotemporal bandwidth consumption profiles. [35], [89].

In essence, next-generation cellular systems and cognitive networks introduce more flexible techniques to react better to these profiles. However, network operators' primary issue is handling and detecting sudden and local abnormal behavior within the network. Whether it is a sharp peak of users' demands (which occurs during mass events, for example), a brief abnormal decrease, or even a non-common daily data consumption pattern.

The first anomaly type needs a fast reaction to guarantee network resilience and service survivability avoiding user rejections. In contrast, the second one may be due to some technical issues of the network infrastructure that need to be repaired instantly. These anomalies are also time-dependent and need geographic identification and temporal detection of the time interval in which they occur with high precision.

3.4.2 Related work

Mobile operators aim to provide many services, including classical voice calls, video gaming, social networking, health care, transport, etc. Hence, providing the best QoS is challenging and requires higher bandwidth, lower latency, and higher speed.

Outliers in a cellular network may occur due to issues such as low coverage areas, sleeping cells, overloading of traffic, etc. They can be linked to unusual events such as social crowding, periodic colossal traffic jams, or unplanned events. Deploying specific cellular facilities for such rare occasions is not an economically viable solution. Yet, operators must cope with this phenomenon and enforce their quality of services (QoS) contracts concerning their clients.

Many mobile network management tasks run autonomously without human intervention. This is because of the evolution of automation and artificial intelligence tools. Hence, detecting issues in the network and solving them become automatic and dynamic without including expert knowledge. New paradigms are introduced in the literature to ease network management tasks. Most proposed schemes employ key performance indicators (KPIs) or machine learning techniques.

The authors in [90] presented an unsupervised clustering technique based on key performance indicators (KPIs). Another anomaly detection framework for KPI time-series data was proposed in [91]. It executes machine-learning regression analysis to detect anomalies.

Reference [92] proposed a spatio-temporal online anomaly detection tool based on Support Vector Machines (SVM) and Support Vector Regression (SVR) to identify regions of interest and detect anomalies. Then, work in [93] also presents a spatiotemporal mathematical model for IoT devices. These approaches applied big data analytics on call detail records (CDRs) datasets.

Hussain et al. [94] applied a semi-supervised machine learning algorithm to detect the anomalies in one-hour data. In [95], they proposed a framework that preprocesses user activities from a real CDRs dataset to create an image-like volume fed to a deep CNN (Convolutional Neural Network) model. In [96], a combination of drones, satellites, and cellular networks is proposed. The paper studies using competitive markets to select the kind of bearer to use. It does not study outlier treatment, nor does it say how different concurrent QoS flows are handled.

The previous studies utilized various traditional machine-learning techniques for outlier detection in a cellular network. Furthermore, some schemes need to provide online anomaly detection, nor do they analyze and diagnose the causes of anomalies. In general, the diagnosis is a complex task and requires good knowledge of the network architecture and services.

More extensive analysis shows that many techniques address different problem formulations of outlier detection. Most of the existing research focuses on one of the following problem formulation[69]:

- ***Sequence-Based approach:*** consists of detecting anomalous sequences from a dataset of test sequences.

- ***Subsequence-Based approach:*** intends to identify anomalous subsequences within a huge sequence.

- ***Pattern-based approach:*** detects patterns in a test sequence with anomalous frequency of occurrence.

- ***Contextual anomaly detection approach:*** detects a group of points or periods that are anomalous regarding their normal behavior.

In this scope, we propose an unsupervised anomaly detection framework that learns autonomously from the network traffic. Detects anomalies in real-time and then executes optimization algorithms for the QoS management and a Spatiotemporal anomaly detection mechanism. The objective is

to minimize network outages and system downtime with the least amount of human intervention.

3.4.3 Adaptive Range-based LSTM Prediction Scheme

We presents in this section a framework based on the data analysis concept to automate the management of resources in cellular networks. Three processes are defined: identifying and detecting anomalies, analyzing the causes, and triggering adequate recovery actions.

First, the proposed solution executes Deep Learning algorithms to forecast the normal behavior of the network and defines dynamic thresholds. Second, it identifies cells with peak demands and raises alarms if the measured real-time data exceeds the threshold values. Third, we define QoS optimization methods to proceed with suitable design for resource allocation and fault detection and avoidance.

In essence, we propose heuristics for QoS management that initiate the deployment of the minimal number of required UAVs (Unmanned Aerial Vehicles) as flying base stations to collect extra data in cells presenting peaks.

In order to optimize network resources, it is essential to detect and successfully remove the anomalies. We consider in our study a real-time-series dataset that we present earlier: CDRs (Call Detail Records) of Milan city. Personal users' data are removed to preserve privacy. Hence, the raw data contains five user-specific activity features: incoming and outgoing SMS, incoming and outgoing calls, Internet usage, and geographical location (Cell ID).

The internet activity in these cells is depicted in Fig.2.26. It is clear that in the last weekend of November 2013 (24^{th}-25^{th}), the traffic load decreased at the university and increased in Duomo square, one of the most attractive touristic places in Milan. Furthermore, end-of-year festivities present some anomalous patterns that occurred during the last week of the year (Christmas celebrations; 24^{th}-25^{th} December (Fig. 2.b)) in all cells. Another annotated anomalous traffic activity is detected at the university from 20^{th} Dec. 2013 correlates with the end of the year vacations.

Network model

Typically, mobile users' activities are different at a particular time and cell. The global view of the proposed framework architecture is illustrated in Figure 3.8. We consider a centralized cellular infrastructure composed of a coordinator base station and base stations (eNodeB). The anomaly detection is distributed where each eNodeB executes the LSTM (Long Short-Term Memory) prediction algorithm, analyzes the CDRs in real-time, and finally calculates required resources for the QoS management if an anomaly is occurred (depicted by red color).

Upon detection of the abnormal cell(s), each BS communicates the anomalous cell ID(s) with the coordinator to initiate remedial actions by launching the deployment of Unmanned Aerial Vehicles (UAVs) having a mission of collecting data in overloaded cells. Using UAVs as flying base stations is an

emerging solution to support the cellular network because of their mobility, flexibility, and adaptive altitude [97].

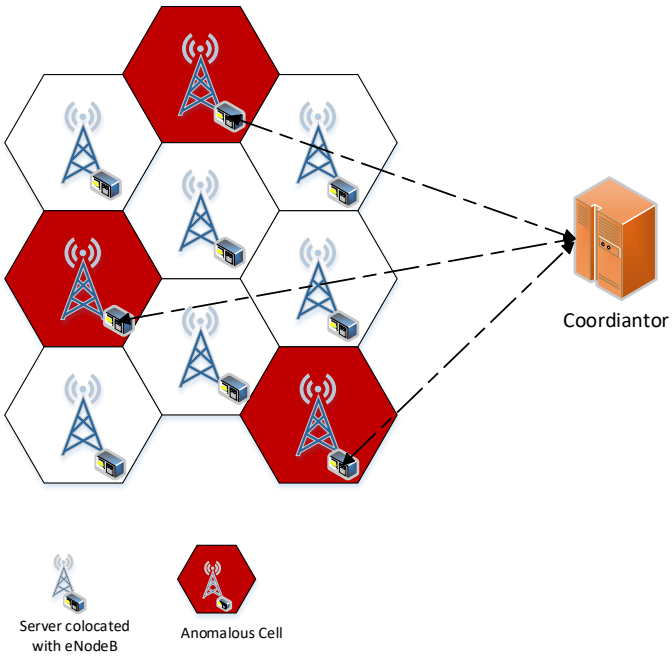


FIGURE 3.8: System topology

Adaptive anomaly detection solution

The anomaly detection solution executes LSTM to train and predict the internet load on each cell. The input data is the set of daily evolution of each user for the presented cell. Figure 3.9 shows the main steps of the dynamic anomaly detection framework, which runs in real-time and compares the incoming sample to the predicted values calculated in the training phase by the LSTM (Long Short-Term Memory).

In essence, the system takes the time-series data as input and then predicts the normal values of users' activity in each cell. The outputs are the minimum and the maximum thresholds of expected values. An alarm is generated if the measured real-time network load is higher or lower than the predicted threshold load values. Once an anomaly is detected, the diagnosis could be automatically triggered. In fact, the QoS management heuristics are executed, and the required number of drones is calculated based on the optimization scheme. Finally, these values are communicated to the coordinator in order to perform the management solution.

We also evaluated our adaptive anomaly detection on a particular event: a protest in the center of Milan. We added semi-synthetic protesting data to the CDR dataset [98]. During a protest in the center of Milan, the amount of uploaded data (photos, videos) to social media could cause sudden congestion to the cellular network and then decrease its quality of service. The

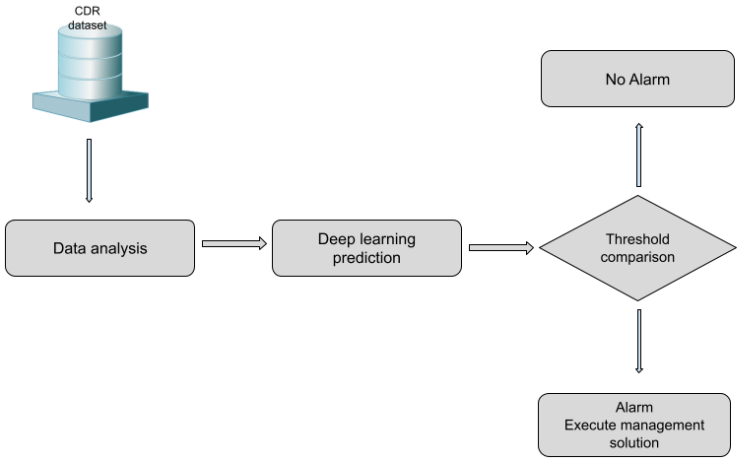


FIGURE 3.9: General flow diagram for the algorithm

prediction model is applied to calculate the typical load of each terrestrial base station as a function of time. Based on this prediction model, we define the minimum (*Min*) and the maximum (*Max*) acceptable threshold values.

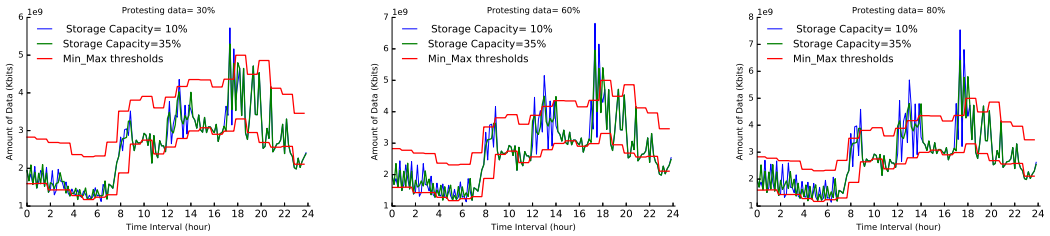
The real-time collected data generated by users' demand is then compared to these predicted values (*Min-Max*). An anomalous time interval is detected if measured real-time data is lower or higher than the appropriate tolerance thresholds. Hence, the terrestrial base stations may fail to handle all connected users because of the congestion within the cell and conduct to a malfunction in the infrastructure. For this reason, we are proposing an automated platform that allows network operators to detect anomalous network cells and launches the deployment of drones to collect data in overloaded cells. The objective is to improve the network's quality of service (QoS)[99].

Experiments

We define a QoS scenario with Two Traffic classes and a limited buffer: The first class is real-time/critical traffic, which is served without delay and corresponds in our dataset to cellular calls. The second is a variable bit rate demand that could be served after buffering. The dataset corresponds to SMS and IoT data. In this section, we are going to focus on the second traffic category.

In fact, data from this class may be delayed using a smoothing buffer. However, the delay should not exceed τ time slots. If cellular bandwidth is available, buffered data can be served on a first-come, first-serve basis. However, if the maximum buffering delay is reached while there is insufficient cellular capacity, drones will be dispatched to provide the required service and meet the maximum delay requirements.

Figure 3.10 presents the network's anomalous behavior for one of these six cells when adding the semi-synthetic protesting data. We study the impact of the data capacity storage or NRT data and vary the percentage of protesting data. We consider 30% (Figure 3.11), 60% (Figure 3.10b) and 80% (Figure 3.10c) of network traffic. Red curves present the range: Min and Max



(A) Protesting data=30% (B) Protesting data=60% (C) Protesting data=80%

FIGURE 3.10: Network load for storage capacity = 35% and capacity =10%.

threshold values of normal traffic calculated based on the LSTM algorithm. Blue and green curves illustrate real-time network load when fixing the storage capacity to 10% and to 35%, respectively.

It is clear that the extra data does not significantly impact the network, with 30% of users' demand for both cases of storage capacity. In essence, we depict an overrated data peak (between 5 PM and 6 PM in Figure 3.11). These load peaks impact the radio channel occupancy causing the anomalies. They are detected if the real-time network load is higher than the maximum value of the predicted data. When the users' demand increases to 60%, we still have the same load peak as the first scenario but with a higher amplitude.

We notice another smaller peak detected between 1 PM and 2 PM. This peak is more important with 10% of capacity storage. This is because data is sent in real-time to the network. However, in the case of 35%, data is stored on the smartphone. Finally, when protesting data reaches 80% of data, the cellular network is seriously impacted in both cases. Indeed, the streaming data causes three peaks (around 9 AM, 2 PM, and 6 PM) with different degrees, but the most important load peak is between 5 PM and 6 PM, where the global data traffic is nearly double compared to the common network measurements.

3.4.4 Spatio-Temporal Anomaly Detection Mechanism (STAD)

Spatio-temporal data is extremely common in many problem settings where collecting data from various spatial locations for the nature of the problem is important. Outlier analysis is an important research area in data mining and machine learning communities. The main objective of this method is to detect outliers in cellular networks using a hybrid deep learning framework composed of two components and evaluated using real datasets of Call Detail Records (CDRs)[69].

We continued and enhanced the work in collaboration, and this section presents the contribution that we have made to work. In [70], a large part of the thesis deals with anomaly detection in TS. The author has extensively used tools based on support vector machines and older mechanisms.

The framework learns spatial and temporal contexts separately and uses those representations to identify spatiotemporal anomalies. It is a double-stage technique that allows network operators to detect spatial and temporal

anomalies. Based on the One-Class SVM (OCSVM) algorithm, the first stage can detect the cells presenting an anomaly, thus its geographical location. The second stage has two alternative solutions based on Long Short Term Memory (LTSM) and Support Vector Regression (SVR) prediction algorithms.

The first component is a set of machine learning classifiers to extract the spatial context from the input datasets. The second component includes several prediction tools, including deep learning, that are trained to predict the temporal context from the same datasets.

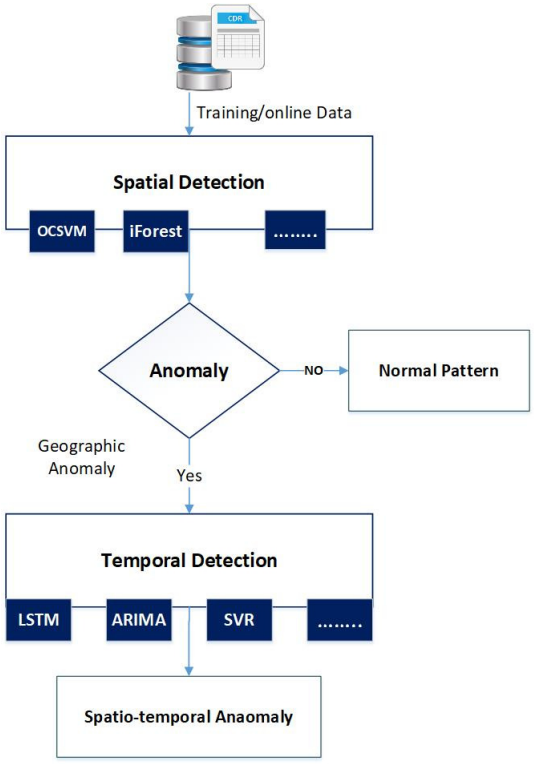


FIGURE 3.11: STAD: Spatio-Temporal anomaly detection framework scheme

We used two datasets in our experiments: the Telecom-Italia dataset. We focus in our performance study on two use-case testbed studies:

- San Siro stadium use-case represents an anomalous consumption peak for some hours during the weekends. The time-series dataset used for validation contains 4,320 data points, of which 72 are anomalous. We discovered some abnormal behaviors within the geographical square where the San Siro stadium is located. In fact, these anomalies consist of a considerable increase in users' network activity with a time-limited peak of SMSs, calls, or even Internet packets.

- The end-of-year festivities use-case contains anomalous patterns that occurred during the last week of the year (from Christmas to New Year's eve). The dataset used for this use case contains a time series of users' activity in Duomo square, one of Milan's most attractive touristic places. The dataset comprises almost 42,000 data points, of which 5,184

are anomalous. We discovered another type of anomaly is related to New Year's eve, with a brief peak of network activities during the first hour of 2014 and during the 1st of January with a decrease of network activities throughout the day. Figure 3.12 depicts the number of calls during November (top figure), December, and the 1st of January 2014 (bottom figure) in the touristic Duomo square. We notice clearly from these two figures the anomalous call amount patterns that occurred on the days mentioned and highlighted with red curves.

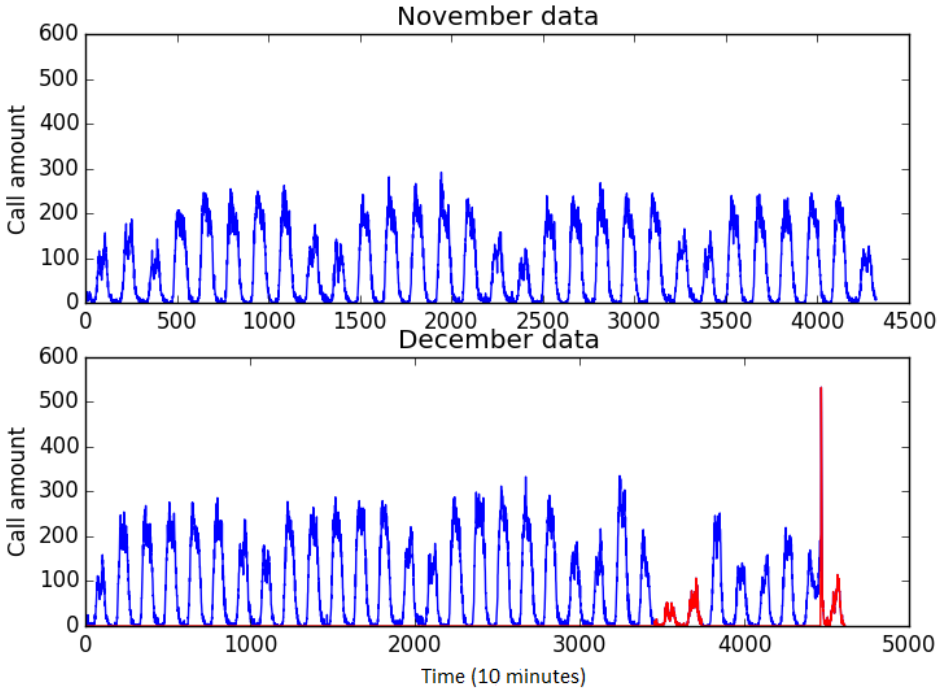


FIGURE 3.12: Duomo square: evolution of numbers of calls)

The second dataset is extracted from the *D4D-Senegal challenge* project [68]. This dataset collects call detail records of phone calls and SMSs of about 9 million users during 2013. It is divided into three components: the first set contains the antenna-to-antenna traffic for 1,666 antennas hourly. The second set contains one year of coarse-grained mobility data at district levels. The third set contains fine-grained mobility data on a rolling 2-week basis for a year with bandicoot behavioral indicators at individual levels for about 300,000 randomly sampled users.

We extracted from the last set the information allowing us to monitor the instant number of users in each cell and, thus, estimate the instant BS load (or users' occupancy) for the whole day. Unlike the previous dataset, D4D CDRs set provides user communication information at a fine-grained space scale, i.e., at each base station instead of geographic square aggregation. Hence, we can monitor each cell apart and detect network outliers precisely.

As with the *Italia Telecom* data, we extracted from these CDRs the times-series data corresponding to Dakar city, which describes the daily evolution

of user occupancy within a BS. As with the Milan data-set use-case, we also focus here on two use-case testbed studies:

- Friday noon use-case: is an anomalous decrease of consumption during some hours. The time_series dataset used to validate our framework with this testbed contains almost 17,000 data points, of which 5,760 are anomalous.

- Tuesday the 5th of February use-case: is a sudden decrease in user consumption at night due to a technical problem. The dataset used for validation contains almost 43,000 data points, of which 720 are anomalous.

The study is more refined for the Dakar use case since the dataset is more important. Figure 3.13 presents some examples of Dakar outlier patterns. A local decrease of user numbers in the cell is noticed for some BSs, which corresponds to one hour, usually between 12 PM and 1 PM(top red curve in figure 3.13). Compared to other days (top blue curves in figure 9), these untypical behaviors usually occur on Fridays. Thus, we consider workday data, except Fridays, to train our first stage part, which allows us to detect these types of outliers automatically.

Another type of anomaly (bottom graph of figure 3.13) is detected by the proposed algorithm. We noticed that on Tuesday, February 5th, the BS activity decreased heavily between 10 PM and 11 PM; this was the same for all BSs installed in Dakar. Otherwise, this sudden decrease is neither present on days of the same week nor other Tuesdays. This type of anomaly may occur due to technical incidents in the network equipment or electricity issues. This anomaly is detected by training our model with the history of Tuesdays' data before February 5th.

Temporal anomaly detection results

In this section, we evaluate STAD's second stage, which is temporal anomaly detection based on SVR and LSTM.

The training procedure is based on calculating the difference between training data and predicted data. This step can be evaluated with several metrics. The most known are: The Root Mean Square Error (RMSE), Mean Square Error (MSE), and R2. Consequently, in our work [33], we compared these error evaluation methods when training algorithms with given datasets. The outcome of the training evaluation for LSTM and SVR is studied. It is seen that MSE is better (see Table 3.8). We also decided to maintain MSE for the classification to have a homogeneous approach.

As explained before, SVR can be advantageous in the case of small amounts of data because of the use of support vectors. However, compared to recurrent neural networks, the general result shows that LSTM architecture overcomes SVR.

Fig. 3.14 presents the predicted values of SVR and LSTM for the Dakar Dataset. Our LSTM architecture gives better prediction values than SVR. For

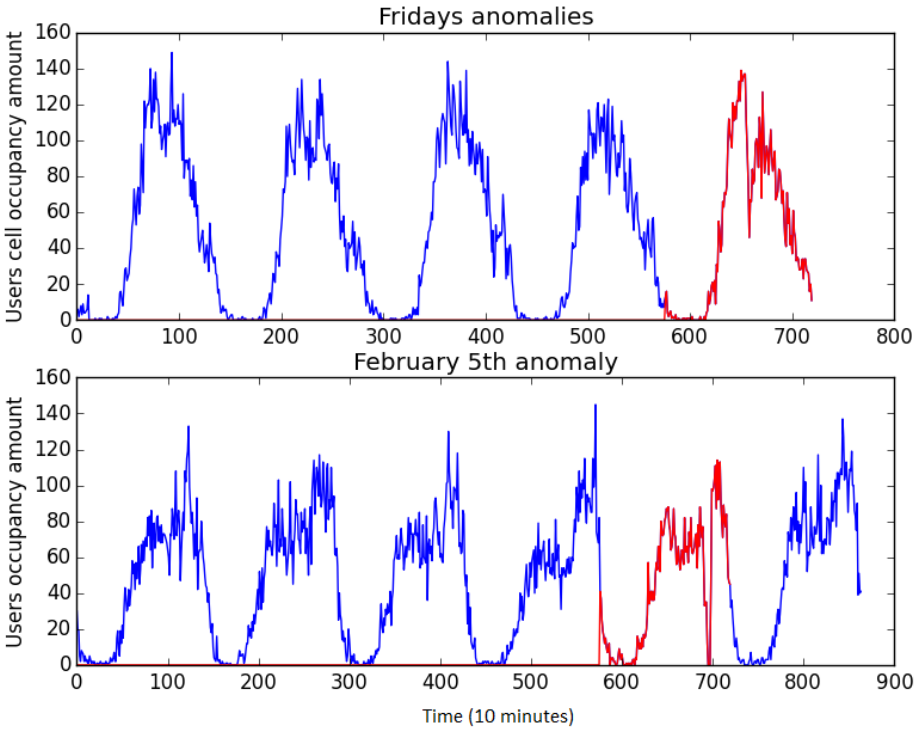


FIGURE 3.13: Examples of Dakar anomalies: the top figure gives an example of a Friday anomaly (red curve) and its previous workdays' normal data (blue curves). The bottom figure shows examples of BS anomalies on February 5th (red curve) and other normal data from Tuesdays.

TABLE 3.8: Metrics comparison

KPI/Algorithms	**LSTM**	SVR
MSE	0.006	0.0175
RMSE	0.063	0.073
R2	0.918	-10.1753

this reason, we define the dynamic thresholds based on the LSTM predicted values. It is clear that the predicted values of LSTM are closer to the real data than the SVR predicted values. The difference in the MSE values calculated over the regression is given hereafter:

TABLE 3.9

Algorithm/MSE	**Value**
SVR MSE	0.03799
LSTM MSE	0.01397

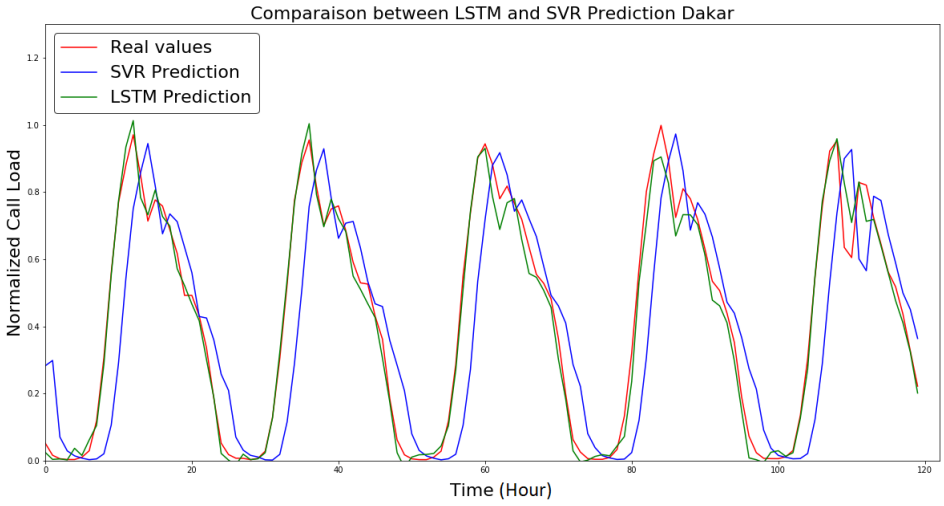
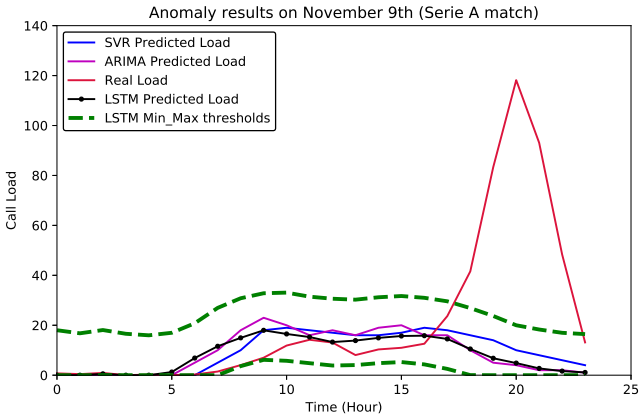


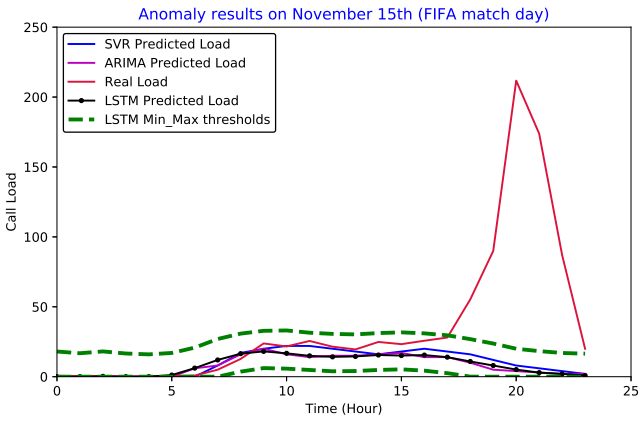
FIGURE 3.14: Comparison between the predicted values of SVR and LSTM for Dakar dataset

Figures 3.15 and 3.16 present the comparison results of temporal anomaly detection for both datasets, the Milan and Dakar cellular networks. Green curves present the range: MIN and MAX predicted threshold values calculated by the LSTM algorithm. The red curves illustrate the real-time network call load. The outliers are detected if the real-time network traffic is higher or lower than the predicted data's maximum or minimum values, respectively.

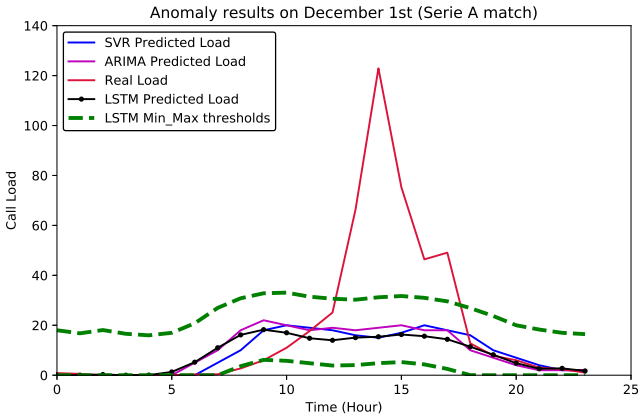
Figure 3.15 shows some abnormal patterns in the San Siro network activity. We noticed from these figures that the anomalous time-interval behavior



(A) Anomaly results on November 9th (Serie A match)



(B) Anomaly results on November 9th (Serie A match)



(C) Anomaly results on December 1st (Serie A match)

FIGURE 3.15: Temporal Anomaly detection result for the San-Siro testbed

detected by the SVR and LSTM is synchronized with the untypical peak of network activity. At the same time, the ARIMA-based model is less accurate.

It is clear that LSTM detects the outlier before other algorithms. In Figure 3.15.a 3.15.b and 3.15.c, we noticed that the ARIMA-based model reports a false anomaly around 10 PM, contrary to SVR and LSTM, due to their overrated time-series prediction at this time-stamp. The call load peak is detected between 5 PM and 10 PM in Figure 3.15.a 3.15.b and between 11 AM and 6 PM in figure 3.15.c. It corresponds to the time of the football matches in the San-Siro stadium. In addition, in figure 3.15.b, we depict that the peak is more important on the FIFA match day and that the ARIMA model starts signaling the anomalous pattern before its exact occurring time, while the SVR and LSTM are well synchronized. The load peak impacts the radio channel occupancy, thus causing anomalies.

Figure 3.16 presents some examples of Dakar's Friday anomalies. We can clearly notice that the detected anomalous time interval is synchronized with the time and the duration when the abnormal decrease of BS users' occupancy occurred. In figure 3.16.c, the detected temporal anomaly takes longer than on other days (usually for almost one hour) and covers the daily activity time. This is because of a special event in that area which caused a sudden decrease in users.

On the other hand, the ARIMA-based model fails to detect the anomalous time window for all examples. Instead of detecting the drastic decrease of cell load occurring between 1 PM and 4 PM, it detected two false anomalies: the first around 11 AM and the second around 5 PM. However, it is clear that LSTM and SVR detect the anomaly because the network call load is lower than the predicted values. Then, LSTM gives better results than SVR.

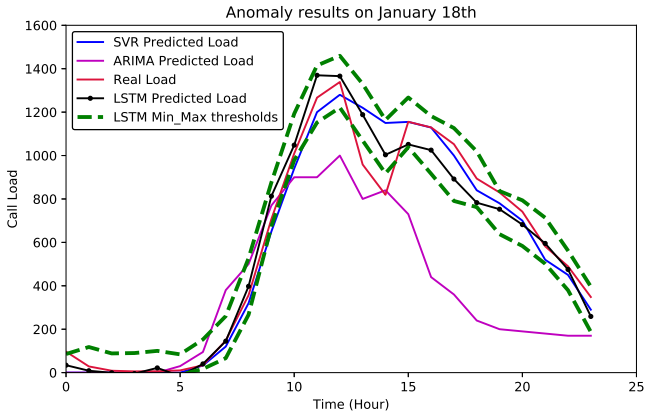
We conclude from these results that the number of subscribers impacts the normal traffic cellular network and drastically impacts the network in some configurations. In fact, it can affect the radio channel occupancy and cause anomalies (case of the San_Siro stadium). This study allows network operators to detect anomalous behavior with a dynamic and adaptive algorithm to improve the network's quality of service (QoS).

3.5 Conclusion

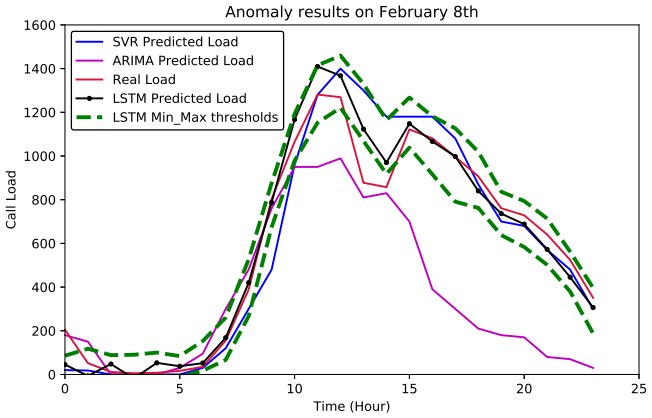
In this chapter, we exploit a more advanced feature that can be found in RNN, and we propose novel applications that can benefit from RNN characteristics. We have proposed deep-learning techniques. (specifically recurrent neural networks) to learn data patterns emanating from typical IoT devices. In a LORA context, with very low available bandwidth, our method helps in drastically reducing raw IoT data transmission and replacing it with the learned neural network parameters (weights).

Weights and periodic samples are sent to the cloud network through LORA, and data is reproduced using the reverse process. As IoT devices can be used to detect anomalies, we adopt a similar neural network to make outlier detection. An alarm is directly sent over LORA in that situation.

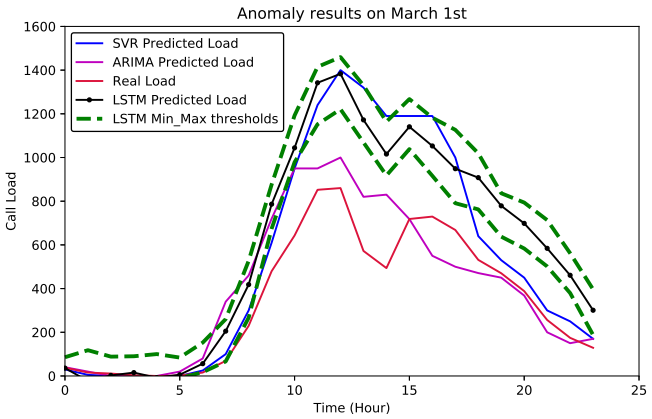
We then proposed a novel transfer learning architecture for time series prediction in modern cellular networks. Intra-class and inter-class transfer



(A) Anomaly results on January 18th



(B) Anomaly results on February 8th



(C) Anomaly results on March 1st

FIGURE 3.16: Temporal Friday Anomaly detection results for the Dakar testbed

learning is applied to those cells in two case studies. The first is when there is a shortage of training data. The second corresponds to the will to personalize

prediction on new networks based on legacy prediction tools. We show that the architecture gives excellent results for predicting traffic load in both cases.

Since network anomalies or outliers are abnormal behaviors that occur suddenly, affecting the performance of mobile networks and causing considerable losses to cellular operators. We proposed a hybrid framework was proposed to detect Spatiotemporal anomalies of multi-variate datasets in an unsupervised way. The proposed anomaly detection framework can be applied to any dataset of users' activity times series.

In fact, it provides automatic and online detection of network anomalies to prevent the issues mentioned above and allows network operators to monitor their infrastructure. The aim is to help them to cope with new challenges in next-generation networks. The STAD framework is executed in two stages and learns temporal and spatial contexts separately and uses those representations to identify spatiotemporal anomalies. The first component is an OCSVM classifier to extract the spatial context from a given dataset. The second component is the range-based LSTM and SVR machine learning algorithms which are trained to remove temporal anomalies.

Chapter 4

Resource Management and optimization Algorithm

4.1 Introduction

The progress in the Internet of Things (IoT) and mobile communications applications produce extensive data that can be collected and managed. Furthermore, resource administration and management dilemmas are omnipresent: in power systems, cellular networks, quality of service (QoS), and many others. In this chapter, we investigate the automation of energy management as the first case and the automatic assistance of a drone to a base station in distress as the second case.

The need to study this automation of electrical resources is mainly due to evident economic and ecological issues. The growing interest in green energies, the implementation, and the use of renewable energy resources (Photovoltaic Panels (PV), wind turbines, and hydroelectric power) are becoming urgent.

These renewable energy resources require the design of mechanisms that will allow better management and waste reduction. In the following, we will mainly be interested in the energy produced with the photovoltaic panels with data in our possession emanating from a real deployed microgrid powered by solar panels. We propose a novel neural network method for managing microgrids in this work. The investigated microgrid is composed of a group of buildings (2 different consumers) that integrate renewable energy production (producers) and batteries (energy storage systems).

We will describe it in more detail in the next part. Abundant information is required to implement an efficient/optimal energy management system (EMS) in a microgrid. The first requirement is the predicted energy consumption that depends on the building and the user's habits.

For the energy storage system, the needed information concerns the capacity of the batteries, the current state of charge (SoC), and the predicted SoC. The third important information required by an EMS is the predicted energy produced by PV panels. It is highly dependent on weather conditions and production and faces a lot of variations.

The architecture of cellular networks demands high levels of optimization to prevent network anomalies from happening. Out-of-range communications, obstacles, unusual human presence, etc., typically cause irregularities.

Because this is one of the most critical issues that can pose service degradation and directly affect the different network functionalities, it is a central problem that we target in the second part of this chapter.

We need to use sophisticated prediction models and optimization methods to find the best management strategy. For a microgrid, the proposed model is based on a novel neural network approach that optimizes electricity usage while assuring customer comfort [100].

For that purpose, we introduced different use cases with various objective functions and constraints with incremental complexity. We first resolved initial use cases with Integer Linear Programming (ILP). Other more complex problems require a heuristic resolution like the Rule-Based Resolution because they are non-linear. Then we propose the use of machine learning models. The first one is the QLearning approach[101]. The second is a recurrent neural network that is then trained with those results and replaces the above two methods as a reinforcement learning system.

The second proposed deep learning solution does not follow the general reinforcement trend using state/action/reward exploration. In this method, we are using a classical deep recurrent neural network. The long-short-term gates will catch the relation between optimal states, actions, and rewards. The heart of our solution resides in learning 'good actions' directly rather than exploring a considerable space of potential solutions, eventually converging (as most deep RL methods do).

Regarding the second use case, we propose to use UAVs on an on-demand basis for offloading base stations to provide services to connected devices. We first suggest an optimal model aiming to maximize the data collection by considering the UAV battery [102]. Then we define a deep reinforcement learning environment based on two techniques: Q-learning and LSTM. They learn the optimal model. After that, we present a comparison study for both proposed approaches based on real cellular network datasets, where we analyze the exactness and convergence times.

4.1.1 Related work

In this chapter, we investigate 1) the prediction of consumption and renewable energy production and 2) the response to the need to serve the demands of end-users.

In [103], the authors are treating how to handle and support all mobile service users with various Quality of Service (QoS) requirements. For that purpose, they are dealing with splitting the network into slices from different properties to provide heterogeneous network demands. They also study the impact of the slicing by using a deep reinforcement learning (DRL) algorithm on the speed of content of user demands. Since the problem of resource management is omnipresent in different disciplines but uses the same approaches, we will focus on resource management for the microgrid.

The authors in [104] proposed a (DRL) algorithm for indoor and home hot water temperature regulators, aiming to decrease energy consumption

by optimizing the usage of PV energy production. In [105], the authors proposed a demand response method to minimize energy utilization from peak periods when the electricity price is higher to off-peak periods when the electricity price is low.

A Q-Learning method is developed to deal with the dynamic electricity prices and different power consumption without compromising the users' comfort. The author also used fuzzy reasoning to model human thinking and evaluate the random action the agent could take as a reward function. Furthermore, an optimized Reinforcement Learning method combined with a Decision Tree method has been proposed in [106]. The aim is to let the agent learn how to manage the power dispatch efficiently into a microgrid environment simulated for a long time.

Hepeng et al. [107] propose a home energy management method based on DRL for optimal scheduling of home appliances. They started by formulating the home energy management problem as a Markov Decision Process (MDP) regarding the randomness of real-time electricity prices and residents' activities. The chosen policy was Proximal Policy Optimization (PPO) which determines the optimal DR scheduling strategy.

As extensively explained in [108], the Reinforcement Learning technique (RL) aims to optimize systems that react to situations by finding heuristically a set of optimal actions for different possible states of the environment to maximize an objective. State-Action-Reward is the general nomination of this family of methods. Further, deep reinforcement learning brings continuity to RL when states are continuous or too large to be solved quickly by RL.

[109] The authors conceived an optimal energy management algorithm based on deep deterministic policy gradients (DDPGs). The proposed model provides the optimal scheduling decisions for Energy Storage Systems (ESS) charging/discharging power and heating, ventilation, and air conditioning systems (HVAC) input power based on the current information.

In [104], a deep reinforcement learning (DRL) algorithm for indoor and residential hot water temperature control is developed, intending to reduce energy consumption while optimizing the usage of renewable energy production. In another work, [110], authors review the use of reinforcement learning for demand response applications in the smart grid. They presented various RL applications. For deep RL, several neural networks have been tailored to reproduce the weight matrix of the system. They have been utilized to control diverse energy systems such as electric vehicles, HVAC systems, smart appliances, or batteries.

Deep Q-Network (DQN) and Policy gradient (PG) [111] are the primary roots from which a large set of derived algorithms have been proposed. Without going too much into a survey of such methods, it is essential to state the main differences between all these techniques and hence introduce our new solution:

- ILP techniques give the optimal solution. However, one quickly reaches limitations: 1 - related to the resolution time, 2 - related to solving problems that can soon become non-linear.

- ILP techniques use a solver for each new dataset. They hence take time and cost money. It comes with a license.
- Deep RL is a heuristic system that can quickly solve problems faster than exact solvers.
- Deep RL methods [112] require a large amount of data to set the weights of the system.

The International Electrotechnical Commission in the standard IEC 61970, related to EMS application program interface in power systems management defines an EMS as "a computer system comprising a software platform providing basic support services and a set of applications providing the functionality needed for the effective operation of electrical generation and transmission facilities to assure adequate security of energy supply at minimum cost" [113].

Two market strategies are proposed and solved using the sequential quadratic programming method. In [114], The authors try to optimize the Microgrid (MG) performance during interconnected operations. Luu Ngoc An et al. in [115] suggested dynamic programming to optimize the process of a grid-connected MG.

The main objectives are reducing the electricity supplier's selling cost and battery aging cost. [116] describes the different methods used by the EMS. They classified them into Classical methods (linear and nonlinear programming, dynamic programming, Rule-Based methods), meta-heuristic approaches (genetic and swarm optimization), Artificial intelligent methods, Stochastic and robust programming approaches, and model predictive control.

The authors [117] propose a home energy management optimization strategy based on deep Q-learning (DQN) and double deep Q-learning (DDQN) to perform the scheduling of home energy appliances. Results are compared with those of Particle Swarm Optimization (PSO) to validate the performance of the proposed algorithm. Results show that the DDQN is more appropriate for minimizing the cost in a HEMS.

Bin Xu et al. [118] integrated a Q-learning algorithm in the energy management system for parallel hybrid electric vehicles. The conducted study aims to determine which factors (learning experience selection, number of states selection, states and action discretization, and exploration and exploitation) are the most influential in an RL system. The results showed that the learning experience selection and state/ action discretization had a higher impact on the vehicle fuel economy.

Authors in [119] developed a decentralized multiagent system (MASs) for a grid-connected microgrid (MG). They studied electrical and thermal energy flows and a variety of distributed energy resources (DERs) using the RL algorithm to optimize the performance of MG and each autonomous agent. As a result, MASs with DRL have shown elastic management while examining consumer consumption and diminished operating cost.

In [120], authors implemented a new variant of reinforcement learning (RL) method Dyna, namely Dyna-H, for energy management of a series of

hybrid electric tracked vehicles. Results show that Dyna-H realizes faster training speed and lower fuel consumption than traditional DQL. Reinforcement learning needs a lot of data and a lot of computational time [121, 112]. Moreover, too much reinforcement learning can lead to an overload of states, leading to solutions that will not converge. Furthermore, more studies need to be conducted on the impact of state actions and rewards on the environment.

Elena Mocanu et al. [122] studied the use of deep reinforcement learning using two methods, Deep Q-learning and Deep Policy Gradient. The objective is to complete the online optimization of schedules for building energy management systems. The result shows that Deep Policy Gradient performs better than Q-learning to make online scheduling.

Authors in [123] developed a Robust Data Predictive Control framework for Energy Management System (RDPC-EMS). They started by predicting the electricity price using the Outlier-Robust Extreme Learning Machine(OR-ELM). Then, the local controller executes the energy scheduled to maintain the MG supply-demand balance.

After implementing some of these techniques, we have found several common difficulties in adapting to our context: huge exploration time (sometimes even, we don't converge at all). Need of large datasets.

The difficulty we face in this contribution comes from two simultaneous problems: First, Mixed Integer Linear Programming (MILP) methods cannot be used in an IIoT environment as explained above (Time and cost). Second, RL using Q-learning, DQN, or PG require a lot of data and considerable training intervals that cannot be available in real life. So the reinforcement learning that we target learns from good experience rather than exploiting a vast (almost infinite) space of state/actions/rewards.

4.2 Energy Use case

4.2.1 Context and problem definition

We are considering different actors of a microgrid that need to collaborate. These actors are the Consumers (Buildings, Houses...), the Renewable Resources Producers (Photovoltaic panels, wind turbines...), The Electricity Supplier, and the Storage System (Battey, Electric Vehicles...). The used architecture is described in Fig. 4.1. It is implemented in two buildings in the Institut Polytechnique de Paris (Drahi building and U4 Evry student dorm). A set of solar panels and batteries are installed on the top of the Drahi building.

The collected data architecture is described in 2.5.1. It consists of the detailed consumption (separate consumption categories are measured by IoT devices and sent to a gateway each second) and the produced energy from the photovoltaic panels. The database used to store the multi-variate collected data is InfluxDB. This database is a non-SQL database specialized in the storage of time series. We used Bokeh as a graphical interface to display and manipulate stored data. We extract the needed information from the database and save it into a Comma-separated values (CSV) file.

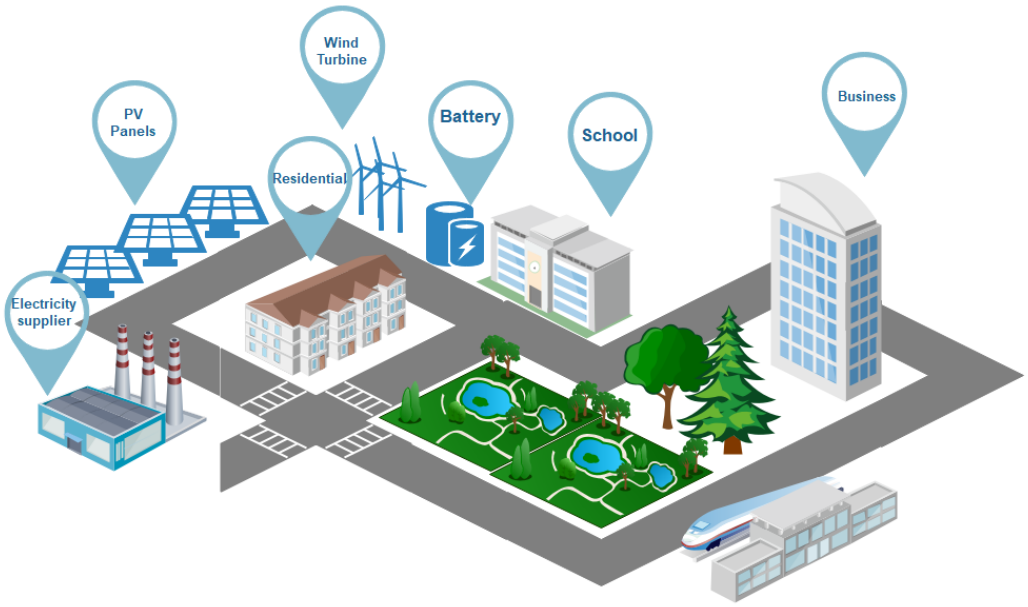


FIGURE 4.1: A Micro Grid Architecture

In our EMS, we have many options. One could want to minimize energy costs, optimize consumer demands, etc... So we naturally have many cases for energy management. The output of our EMS consists of a set of actions that will be applied to the different devices on the system. Depending on the inputs, adequate action will be done according to the various defined variables in table 4.1. The operations could be done on different devices. The EMS could apply the action, use solar production or store the extra produced energy in the batteries.

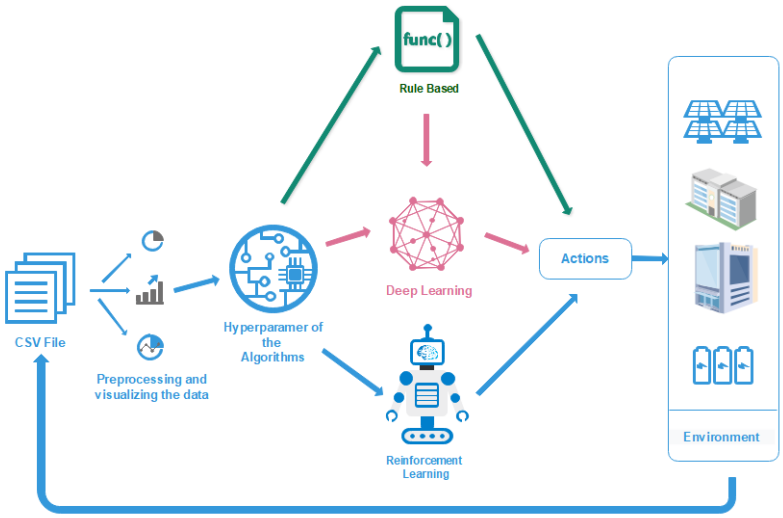


FIGURE 4.2: Microgrid management system Architecture

Figure 4.2 shows the proposed microGrid renewable energy architecture and uses targeting to minimize the purchasing power from the electricity supplier. Consequently, the purpose is to diminish the quantity of energy

procured from the grid and avoid the battery's solicitation to prolong its lifetime. Therefore, at an instant t, we are dealing with three parameters: the consumption, the production of renewable energies, and the battery's state of charge. We are considering an infinite battery size where the extra energy generated needs to be stored. Our objective is to serve the energy demand privileging the use of solar production then, the battery, and finally, the grid supplier.

We want to find at each instant which action we need to apply to our environment. Three different methodologies are compared in this chapter to achieve the defined objective: the Exact resolution, a heuristic resolution, and a machine learning method. For that purpose, we are considering real consumption data collected from the two buildings in the Institute Polytechnique de Paris and real production data collected from Photovoltaic Panels.

The processed data is spread over more than three years. Various features concerning the consumption are available: Date and time (in UTC), weather forecast, the total consumption per zone, etc. Likewise, for production, various information is available Date and time (in UTC), Global_Solar_Flux, and Measured Power at Maximum Power Point. We have several granularity levels, among which we can cite 1 second, 10 min, 30 min, and one hour. The chosen granularity for our experimentation is one hour.

In Table 4.1, we describe the different variables and their definition.

TABLE 4.1: Variables and definition

Variable	Definition
E_t^T	Total energy that needs service in time slot [t-1,t)
E_t^C	Critical energy that needs service in time slot [t-1,t)
E_t^D	Delayed energy that needs service in time slot [t-1,t)
P_t	Produced energy at time slot [t-1,t)
B_t	Estimated Energy in the Battery at time slot [t-1,t)
G_t	Estimated energy drawn from the grid at time slot [t-1,t)
D_t	Total estimated demand of delayed traffic in slot t
R_t	Delayed traffic that has reached its time limit.
M_t	Total delayed traffic generated in $[t-\tau, t)$
ES_T	Total energy saved per day
α_t	Binary, Using Battery energy action.
β_t	Binary, Using grid energy action.
γ_t	Binary, Storing the extra energy to battery action.
ρ_t	Binary, Using solar energy action.
θ_t	Binary, satisfy D_t action.
a_{Tot}	Total actions performed after Tot. slots
τ	How many instant that delayed predicted energy consumption can be shifted

4.2.2 Uses cases

Four different use cases were explored. In the first one, we consider only one energy demand class. The main idea is to use renewable energy production as much as possible. In fact, the purchase from the energy supplier is penalized. We also aim to minimize the use of batteries both at consumption and storage levels. In the second use case, we are considering only one energy consumption type, but this time we aim to store the extra produced energy in the batteries.

For example, in the first case, if we need to serve 5kwh at 10 AM and we produce 6kwh, the extra 1kwh will not be stored, unlike the second case, where this excess production will be held.

In the third case, we categorize energy consumption into delayable and critical. The critical demand needs to be served without any delay. In this case, the delayable request could be served later within any time limit. While in the fourth case, the delayed demand could not exceed a specific time τ.

4.2.3 Exact Resolution

We start with a simple algorithm and progressively get more complex. Since they are intended to run in IIoT distributed microcontrollers, it makes sense to have incremental complexity adapted to different hardware product lines.

Case One

Model Explanation The first case tries to minimize actions in the EMS. The goal is not only to reduce the amount of energy bought from the grid but also to avoid solicitation of the battery to extend its lifetime.

We are considering the energy demand entirety at an instant t. The actions are:

- Using the Battery
- Buying from the electricity supplier
- Using the produced energy at that time t.

Our objective is to minimize the total actions performed to serve the energy demand after T time slot. We have decided to penalize the purchase action β since we aspire to have a self-sufficient network grid. Equation (4.1) ensures that at each time slot t one action will be equal to 1. Eq. (4.2-4.4) guarantees that we are allowed to execute only one action. The State of Charge of the battery is updated for each time slot in equation (4.5) and depends on the anterior values of α, β, and ρ. Equation (4.6) assures that we will complete the Total energy that needs service in time slot E_t^T. Equation (4.7) determines the amount of energy bought from the energy supplier at each time slot.

$$Minimize : a_{tot}=\sum_{t=1}^{T} \alpha_t+penality*\beta_t+\rho_t$$

subject to:

$$\alpha_t+\beta_t+\rho_t==1 \tag{4.1}$$

$$\alpha_t*\beta_t==0 \tag{4.2}$$

$$\alpha_t*\rho_t==0 \tag{4.3}$$

$$\beta_t*\rho_t==0 \tag{4.4}$$

$$B_t=B_{t-1}+(1-\rho_{t-1})P_{t-1}-\alpha_{t-1}E_{t-1}^T \tag{4.5}$$

$$\alpha_t B_t+\rho_t P_t>=(1-\beta_t)E_t^T \tag{4.6}$$

$$G_t=\beta_t*E_t^T \tag{4.7}$$

Case two

Model Explanation The difference between the first and second scenarios comes from the will to store any extra renewable energy that exceeds the total energy required at time t. For that purpose, we introduced a new binary variable *gamma* that will equal one if The production exceeds the demand. Equation (4.9) stipulates that this difference should be positive, and equation (4.8) stores this extra energy in the battery. Note that the case when this difference is negative is considered in both scenarios by ρ. In fact, if ρ is equal to 0 for both equations (4.8) and (4.9), this difference will be zero.

$$Minimize : a_{tot}=\sum_{t=1}^{T} \alpha_t+penality*\beta_t+advantage*\gamma_t+\rho_t$$

subject to: (4.1), (4.2), (4.3), (4.4), (4.6) and (4.7):

$$B_t=B_{t-1}+(1-\rho_{t-1})P_{t-1}+\gamma_{t-1}(P_{t-1}-E^T_{t-1})-\alpha_{t-1}(E^T_{t-1}) \tag{4.8}$$

$$\gamma_t*(P_t-E^T_t)>=0 \tag{4.9}$$

We intend to minimize the energy bought from the electricity supplier for the third and fourth cases.

Case three

Model Explanation We categorized the energy consumption demand in previous work [124]. We identified the first category as the critical energy demand (CE) and the second as the delayed energy demand (DE). The CE demand is the real-time demand that we cannot delay over time. The DE is the demand that we can serve later without any time limit. The distinction between the first two cases and the last two cases is the considered energy demand that needs to be served at an instant t.

We introduced a new binary θ, equal to 1 if we satisfy the total estimated demand of delayed energy in time slot t. Equation (4.10) updates the State of Charge (SOC) of the battery. Equation (4.25) ensures that at an instant t, we will serve the Critical Energy (E^C_t) that needs service in this time slot. Equation (4.26) defines the amount of energy that we need to buy from the energy supplier to satisfy the CE demand for a specific time slot. Equation (4.13) stipulates that the difference between the production and the CE should be positive. Equation (4.14) and (4.15) explain when θ_t must be equal to 1. The update of D^D_t is made according to equation (4.16).

$$Minimize : G_T=\sum_{t=1}^{T} G_t$$

subject to: (4.1), (4.2), (4.3) and (4.4):

$$B_t=B_{t-1}+(1-\rho_{t-1})P_{t-1}+\gamma_{t-1}(P_{t-1}-E^C_{t-1})-\alpha_{t-1}(E^C_{t-1})-\theta_{t-1}*D^D_{t-1} \tag{4.10}$$

$$\alpha_t B_t+\rho_t P_t>=(1-\beta_t)E^C_t \tag{4.11}$$

$$G_t = \beta_t * E_t^C \tag{4.12}$$

$$\gamma_t * (P_t - E_t^C) >= 0 \tag{4.13}$$

$$G_t + B_t + P_t - E_t^C - \theta_t D_t^D >= 0 \tag{4.14}$$

$$G_t >= 0 \tag{4.15}$$

$$D_t^D = (1 - \theta t - 1) * D_{t-1}^D + E_t^D \tag{4.16}$$

Case Four

Model Explanation The DE, in this case, can now be shifted only for the τ time step. After the DE achieves its limit τ, it will be considered CE demand, which needs to be served at that time slot t. The update of the battery is calculated by the equation (4.29). If equation (4.30) and (4.31) are satisfied then θ_t can be equal to 1. The update of D_t^D is made in equation (4.32). We introduce two new variables. M_t is used to keep track of the DE that arrived in the $[t - \tau, t]$ equation (4.21). While R_t is higher or equal to the total delayed demand, D_t^D, less the demand in M_t and the delayed demand that reached its limit $E_{t-\tau}^D$ as evaluated by equation (4.23).

$$Minimize : G_T = \sum_{t=1}^{T} G_t$$

subject to: (4.1), (4.2), (4.3), (4.4), (4.25), (4.26), (4.13) and (4.15):

$$B_t = B_{t-1} + (1 - \rho_{t-1}) P_{t-1} + \gamma_{t-1}(P_{t-1} - E_{t-1}^C) - \alpha_{t-1}(E_{t-1}^C) - R_{t-1}^D - \theta_{t-1} * D_{t-1}^D \tag{4.17}$$

$$G_t + B_t + P_t - E_t^C - R_t^D - \theta_t (D_t^D - R_t^D) >= 0 \tag{4.18}$$

$$D_t^D = (1 - \theta_{t-1})(D_{t-1}^D - R_{t-1}^D) + E_t^D \tag{4.19}$$

$$R_t^D >= 0 \tag{4.20}$$

$$M_t = \sum_{i=t-\tau}^{t} E_i^D \tag{4.21}$$

$$R_t^D <= E_{t-\tau}^D \tag{4.22}$$

$$R_t^D >= D_t^D - (M_t - E_{t-\tau}^D) \tag{4.23}$$

4.2.4 Heuristic Resolution

Bin Packing

The bin packing problem involves storing objects with a minimum number of boxes. The classic problem is defined in one dimension, but there are many variants in two or three dimensions [125].

We are considering the bins that will contain the elements that require placement. We chose it only because it gives an optimal solution. Algorithm 1 is based on the Bin Packing (BP) problem, particularly the heuristic First-Fit. In our case, the bins will be the produced energy, and the elements to place are the energy consumption demand. The policy of the First-Fit is to

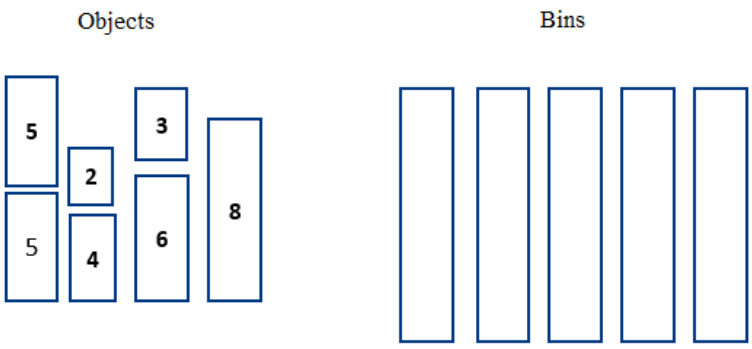


FIGURE 4.3: Bin Packing Problems

browse elements and bins one by one and place the first element in the first suitable bin.

Since the production is variable, the size of our bins will vary as a function of time, so we will be using a Variable Size Bin Packing VSBP [126]. The major limitation of the BP family is that it does not meet time constraints. In fact, in 1, we describe how the heuristic First-Fit deals with the energy demand and place it in the bin.

The algorithm will store the old SoC of the battery and the produced energy in the new bin and try to serve the demand with that bin. If more is needed, it will pass to the next one. The algo2 presents the use of the First-Fit heuristic while considering the third case. In this case, the BP will start with serving the critical demand. Once done, it will serve the delayed one if there is still enough energy in the bin. Dealing with the time-dependent problem, BP is not the best solution, even if it could offer good performance since it does not meet time constraints.

Algorithm 1 Energy Management Algorithm without classification using FirstFit heuristic for BinPacking : EMAWCFF

Input: E_t^T, P_t, B_t length (E_t^T), length (P_t)
Output: G_t,B_t, a_{tot}
Store harvesting energy in the Battery
Increment a_{tot}

while length (E_t^T) **do**

  if $E_t^T \leq B_t$ **then**
    Raise Place Object
    Update B_t
    Increment a_{tot}

  else
    check next bin
    Increment a_{tot} (penalty)
  end if

end while

Algorithm 2 Energy Management Algorithm with classification using FirstFit heuristic for BinPacking

Input: E_t^C , E_t^D , P_t, B_t length (E_t^T), length (P_t)
Output: G_t,B_t, a_{tot}

while Critic Energy Demand **do**
    EMAWCFF(E_t^c, P_t, B_t length (E_t^T), length(P_t)

end while

while Delayed Energy Demand **do**
    EMAWCFF(E_t^D, P_t, B_t length (E_t^T), length (P_t)

end while

Rule Based

Rule-based systems provide the computational mechanisms found in most expert systems. These production rules, in many cases, allow a straightforward encoding of expertise about a particular domain, often as the situation–action pairs where the IF part of the rule specifies aspects of a concern leading to one or more actions as described in the THEN portion [127].

The main idea of a rule-based system is to capture the knowledge of a human expert in a specialized domain and embody it within a computer system. A rule-based system uses rules as the knowledge representation. These rules are coded into the system in the form of if-then-else statements. Hence, the rules are encoded knowledge. Sometimes, rule-based systems are compared to fake intelligence because of the missing learning capability [128].

In this section, we are presenting the developed Rule-Based algorithm for the different defined use cases. We chose to rewrite the above ILP problems in a Rule-Based method since ILP cannot be solved with CPLEX for large datasets and because the last case is non-linear.

Algorithm 3 First case Rule-Based Energy Management Algorithm:FEMA

Input: E_t^T , P_t, B_0, $G0$
Output: $alpha$, $beta$, rho,G_t,B_t, a_{tot}
Update the SoC of the Battery

if $E_t \leq P_t^T$ **then**
    Raise action consume
    Increment a_{tot}
    return 0,0,1,0,B_t , a_{tot}

else if $E_t \leq B_t$ **then**
    Raise action use Battery
    Increment a_{tot}
    return 1,0,0,0,B_t , a_{tot}

else
    Raise action Buy
    $G_t = E_t^T$
    Increment a_{tot}
    return 0,1,0,G_t,B_t , a_{tot}

end if

The second algorithm for energy management will consider the overextended production and store it in the battery as an action.

Algorithm 4 Second case Rule-Based Energy Management Algorithm:SEMA

Input: E_t^T , P_t, B_0, $G0$
Output: $alpha$, $beta$, $gamma$, rho,G_t,B_t, a_{tot}
Update the SoC of the Battery

if $E_t < P_t^T$ **then**
    Raise action consume
    Raise action store
    Increment a_{tot}
    return 0,0,1,1,0,B_t , a_{tot}

else if $E_t == P_t^T$ **then**
    Raise action consume
    Increment a_{tot}
    return 0,0,0,1,0,B_t , a_{tot}

else if $E_t \leq B_t$ **then**
    Raise action use Battery
    Increment a_{tot}
    return 1,0,0,0,B_t , a_{tot}

else
    Raise action Buy
    $G_t = E_t^T$
    Increment a_{tot}
    return 0,1,0,0,G_t,B_t , a_{tot}
end if

The third Algorithm will consider two categories of consumption and will serve them according to their nature and depending on the existing resources.

For the last algorithm, LEMA, we introduced the notion of the deadline to the delayed demand. For example, one could imagine a water heating system or an air conditioner that can be delayed to start later in time but not indefinitely to prevent degradation of the user's comfort.

Rule-based systems face different dilemmas. Rule-based systems also cause other problems. For example, it's challenging to add rules without introducing contradicting rules. The maintenance of these systems then often becomes too time-consuming and costly. That's generally the showstopper for rule-based systems and usually the point where learning systems get into the game.

4.2.5 Machine Learning Method

Since machine learning methods perform well when dealing with optimization problems, we decided to implement two different types of recurrent neural strategies to solve our use cases.

QLearning

Reinforcement learning (RL) is a branch of machine learning that presents an interaction between an agent and the environment, as shown in Figure 4.4. It gives the agent the ability to learn to take action as a function of the reward. The environment is usually assumed stationary, and the state contains the necessary information to determine the best action to perform in the current

Algorithm 5 Third case Rule-Based Energy Management Algorithm:TEMA

Input: E_t^C,E_t^D, P_t, B_0, G_0, D_0
Output: $alpha$, $beta$, $gamma$, rho,$theta$, G_t,B_t, a_{tot}, D_t
Update the SoC of the Battery
Update the Total demand of delayed traffic in slot t

if $E_t^C < P_t^T$ **then**
  if $D_t < P_t^T$-E_t^C **then**
    Raise action Satisfy D_t
    Increment a_{tot}
    return 0,0,1,1,1,0,B_t , a_{tot},D_t
  Raise action consume
  Raise action store
  Increment a_{tot}
  return 0,0,1,1,0,0,B_t , a_{tot},D_t

else if $E_t == P_t^T$ **then**
  Raise action consume
  Increment a_{tot}
  return 0,0,0,1,0,0,B_t , a_{tot},D_t

else if $E_t \leq B_t$ **then**
  if $D_t < P_t^T$ **then**
    Raise action Satisfy D_t
    Increment a_{tot}
    return 1,0,0,0,1,0,B_t , a_{tot},D_t
  Raise action use Battery
  Increment a_{tot}
  return 1,0,0,0,0,0,B_t , a_{tot},D_t

else
  Raise action Buy
  Increment a_{tot}
  $G_t = E_t^T$
  return 0,1,0,0,0,G_t,B_t , a_{tot},D_t
end if

state. Rewards are calculated based on the environment response to actions taken by the agent. The interaction between the agent and its environment defines the mapping state/best action.

In this work, our goal is to design a learning agent able to optimally control the microGrid. We define the tuple (S, A, R) where S, A, and R are the set states, the set of actions, and the reward function, respectively. In the following, we present the different elements of the RL approach.

- **States**: The state contains all the information to choose the best action, which is decisive in the performance. In our context, available solar power output and consumer load are the variables defining the dynamic environment. Furthermore, the level of the battery charge and the grid level must also be considered. In fact, we define the state space based on 4 variables: The energy consumption C, the grid G, the battery level B, and the production P. The energy consumption and the production are inputs; however, the battery level and the grid are calculated based on the chosen action.

- **Actions**: The chosen action defines the next states. The action space considered in this context is:

  . 'B': Using the Battery

  . 'S': Storing extra energy on the Battery

Algorithm 6 Fourth case Rule-Based Energy Management Algorithm:LEMA

Input: E_t^C,E_t^D, P_t, B_0, G_0, τ
Output: *alpha*, *beta*, *gamma*, *rho*,*theta*, G_t,B_t, a_{tot}

for i in lenght (E_t^C) **do**
    Update the Soof the Battery

    if E_t^C[i] $< P_t^T$[i] **then**
        Raise action consume
        Raise action store
        Update P_t*[i]*
        Increment a_{tot}
        return 0,0,1,1,0,0,B_t *,* a_{tot}

    else if E_t[i] $==$ P_t^T[i] **then**
        Raise action consume
        Update P_t*[i]*
        Increment a_{tot}*[i]*
        return 0,0,0,1,0,0,B_t *,* a_{tot}

    else if E_t[i] $\leq$ B_t[i] **then**
        Raise action use Battery
        Update P_t*[i]*
        Increment a_{tot}*[i]*
        return 1,0,0,0,0,0,B_t *,* a_{tot}

    else
        Raise action Buy
        Update P_t*[i]*
        Increment a_{tot}*[i]*
        G_t*[i]* = E_t^T*[i]*
        return 0,1,0,0,0,G_t,B_t *,* a_{tot},D_t
    end if

    if i $> \tau$ **then**
        if E_t^D[i-τ] $<$ P_t[i] **then**
            Update P_t*[i]*
            theta = 1
            forj in range(i-τ**,i)**
                if$E_t^D[j+1] < P_t$**[i] then**
                Update P_t*[i]*
                E_t^D*[j+1] = 0*
                rho = 1
        else if E_t^D[i-τ] $<$ B[i]-*alpha*[i]*E_t^C[i] **then**
            Update B_t*[i]*
            forj in range(i-τ**,i)**
                if$E_t^D[j+1] < P_t$**[i] then**
                Update P_t*[i]*
                E_t^D*[j+1] = 0*
                rho = 1
        else
            beta[i] = 1
            Update G_t*[i]*
            forj in range(i-τ**,i)**
                if$E_t^D[j+1] < P_t$**[i] then**
                Update P_t*[i]*
                E_t^D*[j+1] = 0*
                rho = 1
        end if
    end if
end for

. 'G': Buying grid energy

. 'P': Using the solar-produced energy

For example, if the agent's decision is $'S'$, the battery capacity will be impacted. An hourly action automatically brings the agent in the next

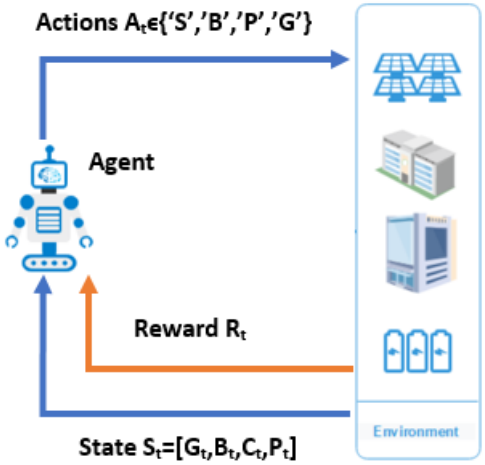


FIGURE 4.4: RL interaction

hour for a new decision.

- **Reward function**: The reward function is the utility the agent receives for performing the right action. It is defined as optimizing energy consumption and storing the overproduction, then minimizing the entire system's operational costs. The reward is presented by a binary value and calculated based on the Chosen action. Since the battery cannot be charged and discharged simultaneously, only one of these actions can be selected and performed at anytime step t.

- **Q-Learning Algorithm**: In [129] a comparison between DQN (Deep Q-Network) and QL (Q-Learning) was made. Results showed that QL leads to better results, and that's why we consider the Q-learning Algorithm, which is a model-free reinforcement learning where the agent explores the environment to find optimal action-selection policy [130].

  In essence, the agent needs to know what the states are and what possible actions are available for each state. It does not need to have any model of the environment. Essentially, the outcome of applying an action to a state depends only on the current action and state. The algorithm defines the optimal policy that assigns a probability distribution to each state on the set of actions. The estimated probability is called Q-Value and is updated based on the received reward after each action.

  The reward formalizes how good it is for the agent to emit a specific action in a given state, considering the optimization of the objective function. It is positive if the agent chooses the right action and negative otherwise. In our case, the optimal policy is a deterministic mapping from the set of states to the set of actions: in each state, there is one best action. Finally, the states are visited infinitely often, and the Q-values are continuously updated till the algorithm becomes convergent.

Another key concept is the Q-Learning table, which presents states by randomly initialized actions. Then each cell is updated through training. In fact, as the agent interacts with its environment, it updates the Q values and learns the optimal policy's Q value. Initially, the agent tries various actions and explores the set of actions, and as it learns, it focuses more and more on the best actions using the "greedy method."

Finally, we will repeat this treatment for many episodes going from the initial state to a terminal state until the learning agent converges toward a good policy.

LSTM

The second used algorithm to deal with the EMS is the LSTM network to train the reinforcement part. RNN is the most appropriate family of NN in our case as it keeps excellent short and long-term memories of events and can correlate them. The DRL architecture is based on a Dense LSTM model. The NN model is composed of LSTM enriched with the 'Dense' property. We add at the end of this RNN one deep layer (fully connected neuron layers), resulting in a complex mathematical architecture that gives much better results than before [131] [21].

The LSTM is composed of 100 neurons and four layers, the number of iterations performed is 4000, and the used optimizer is ADAM. The inputs of the neural networks consist of the different consumption and production data on one hour basis. Since LSTM is a supervised learning algorithm, we will need to feed it with inputs and labels.

The label sets used for backpropagation are the results of the optimization models (daily vectors of $\alpha(t)$, $\beta(t)$, etc...) referenced as "Optimal_ILP_action" in the inputs of the algorithm.During the learning phase, the first step is to analyze the consumption and production data and determine if there is any erroneous or missing data. Data is normalized and split into train and test (70 %, 30 %). After training, we test with one-third of the three-year dataset in our possession to evaluate the preciseness of the actions taken.

So if we summarize the difference between our approach and a classical RL mechanism: First, we directly learn the optimal actions calculated from the model (here it is a Rule-Based Algorithm), whereas the other RL needs to explore all the space, including wrong actions. Second, RNNs have an excellent memory of the correlation between subsequent actions, while plain RL cannot do so. We need to use more complex deep RL architectures to get close to an RNN, such as Duelling architectures.

4.2.6 Performance results for Resource management

In this section, we evaluate the resolution of the four use cases for energy management defined beforehand. We first comment on the results obtained with the ILP resolution. Then the use of the Rule-Based policy, and finally, the use of reinforcement learning. Mainly, we use the following error metrics: mean squared error (MSE), root mean squared error (RMSE), R squared (R^2), as well as the percentage of correct action per day.

The main idea presented in this section is how good the performance of LSTM on the predicted actions is. Even if calculating energy saving is not the primary objective of this work, we added an energy-saving parameter. In the previous submitted work, we compared the use of different methods of Bin Packing, Rule-Based, and Q-learning for the Energy Management System. We also compared the quality of the chosen action for each of the mentioned methods. The principal purpose of this work is to use the acquired information from the exact resolution as input so that we can mimic the exploitation phase of the traditional DRL algorithm.

Since DRL will pass through 2 steps, the first one being the exploration and the second the exploitation, in which it will use the information it learns during the first phase. We intended to emulate this phase by passing the results of the exact resolution as labels of the LSTM algorithm. The percentage of saved energy for one day is equal to 73.8%. The calculation is done according to the following equation:

$$ES_T = (\sum E_t^T - \sum G_t) / sum E_t^T \tag{4.24}$$

Exact resolution

Fig. 4.5 shows the result of solving the first case using CPLEX. This Figure describes the results when varying the battery capacities from 15kwh, 33kwh, 99kwh, and 150kwh to infinite while resolving the first case with the ILP optimization. We can see the optimal results evaluated by CPLEX. 4.5a gives the minimum set of actions found, and 4.5c shows the necessary energy bought from the grid. Fig. 4.5a shows the result from 0 to 23, a day with a granularity of 1 hour. It is the first iteration, with an initial SoC for the battery equal to 10kwh.

The difference between the different battery capacities will be visible in the long term. The number of actions will be incremented for each action carried out. In this case, an action can be (store in the battery, use production or buy from the supplier) the purchase is penalized. The more we perform the purchase action, the higher the number of actions will be.

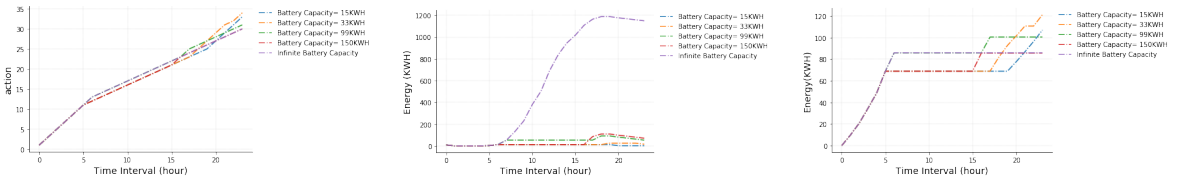
The relation between the number of actions and the battery capacity is the capability of the latter to store to minimize the purchase. We need

to choose a battery capacity that is adequate for production and needs and not that it is incredibly high. In this case, the number of actions required is more significant when the battery capacity is infinite due to priority order over the choice of action. The first action to be performed is the Use of the production. The 2nd is the Storage of the battery, then the Use of the battery, and finally, purchase from the supplier.

Since the battery capacity is one of the variables to consider when defining the ILP problem, the battery capacity will influence the CPLEX resolution and the priority of the actions to be taken. For this reason, in Fig 4.5b, the SoC of the battery when the capacity is 99 kWh is higher between the time intervals of 5 to 10 when the capacity is 150 kWh . What should be noted is that Figures 4.5 (a, b, c) deal with the first case, either we stock or use production.

As the battery capacity will influence the action to be taken, at time point 6 to meet the objective function and satisfy the order of priority of actions, the buy action has been taken. As for the rest of the capacities, another action was chosen. Since curve 4.c presents the Cumulative amount of energy bought, the curve representing the battery's infinite capacity will be greater than the other curves between intervals 5 and 15. Subsequently, we notice that the cumulative amount of energy bought will be lower for batteries with a capacity of 150 kWh and infinite.

4.5b describes the state of charge of the battery.

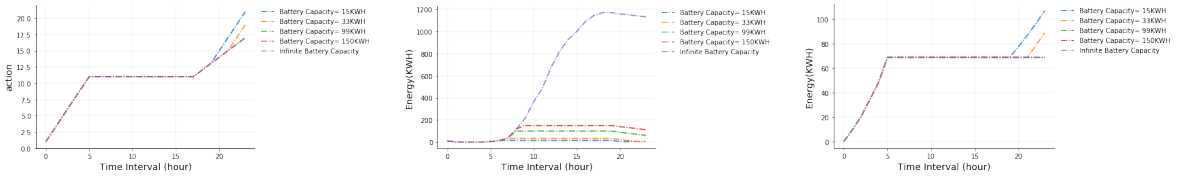


(A) Cumulative action (B) Cumulative SoC of the battery (C) Cumulative amount of energy bought

FIGURE 4.5: First Case CPLEX Resolution for one day

Rule-Based

Similarly, we evaluate Rule-Based algorithms on the predicted dataset. Fig. 4.6 describes the results of varying the battery capacities from 15kwh, 33kwh, 99kwh, and 150kwh to infinite while resolving the second case with Rule-Based algorithms. We made simulations using the Rule-Based algorithm and obtained the result that deals with the four defined use cases. We varied the battery capacities for each use case from 15 kWh to infinite then. Then we plotted the results for the cumulative actions, cumulative SoC of the battery, and the cumulative amount of energy bought. The second case results are shown in Fig. 4.6.

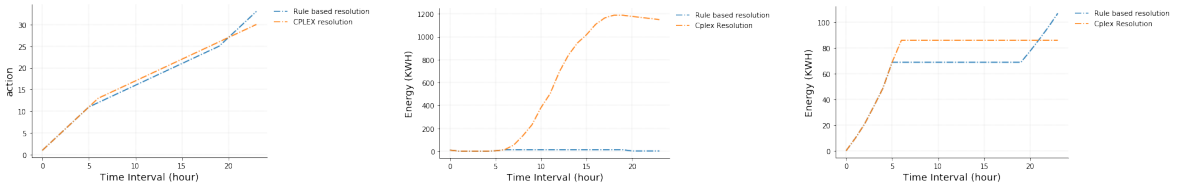


(A) Cumulative action (B) Cumulative SoC of the battery (C) Cumulative amount of energy bought

FIGURE 4.6: Second Case Rule-Based Resolution for one day

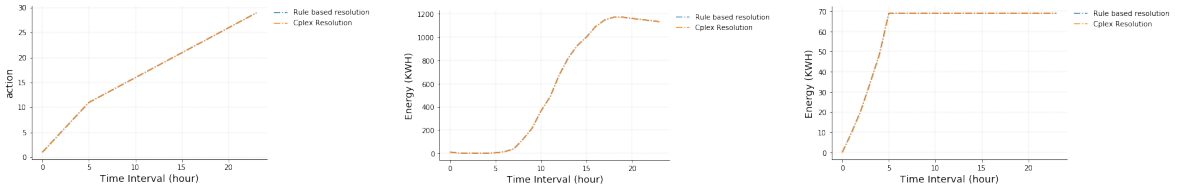
Exact resolution versus Rule-Based

We plot Fig. 4.7 and 4.8 to show that exact ILP and Rule-Based do not give very far optimal results. For the Rule-based approach, the system uses the produced energy and doesn't store it in the battery. For this reason, the SoC during the whole time interval is almost zero. Since the production is at its highest from interval 5, the Cumulative amount of energy bought for the Rule-Based will be lower than for the CPLEX resolution. Thus, it is an incentive to develop less stringent Rule-Based algorithms instead of using limiting ILP solvers.



(A) Cumulative action (B) Cumulative SoC of the battery (C) Cumulative amount of energy bought

FIGURE 4.7: Comparison CPLEX and Rule-Based First Case Resolution

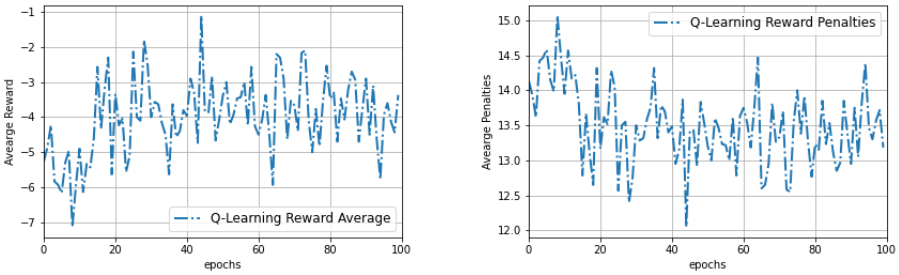


(A) Cumulative action (B) Cumulative SoC of the battery (C) Cumulative amount of energy bought

FIGURE 4.8: Comparison CPLEX and Rule-Based Second Case Resolution

RL evaluation

Results of Q-Learning Approach To measure the performance of the Q-learning algorithm, we consider 6000 episodes. An episode is presented in 24 hours, namely 24 epochs (or steps). In Fig. 4.9, we present



(A) Reward (B) Penalties

FIGURE 4.9: Performance evaluation of Q-learning algorithm

the performance of the Q-Learning algorithm. Fig 4.9a plots the cumulative reward as a function of the number of epochs. It is clear that initially, the agent explores the environment and commits errors. When it has explored enough, it can act wisely, maximizing the rewards when taking the best actions. However, in this case, the algorithm has probably not reached the optimal policy to provide good enough decisions to manage the energy system. More time is needed to learn, and then, the larger the reward means the agent is doing the right thing.

In Fig. 4.9b, we evaluate our agents according to the average number of penalties per episode. The penalties decrease when the agent chooses the wrong action. The smaller the number, the better the performance of our agent. Ideally, we would like this metric to be zero or very close to zero. In essence, a higher average reward would mean that the agent understood and made the right decisions with minor penalties. However, in our case, the agent takes so much time to learn, and both curves start to converge when the number of episodes reaches 100.

Deep Learning vs Q-learning In this section, we will present the different results while comparing the two proposed approaches: Deep Learning and Q-Learning. In Table 4.2, we compare the execution time for both LSTM and QLearning. For this comparison, we used one year of consumption and production data. We then varied the number of iterations from 5 to 100.

The obtained results show that the execution time of LSTM is less important than the execution time for Q-Learning. In fact, for 100 iterations, LSTM will complete the training step in 17.16 seconds, while it would take more than 75 hours for Q-Learning. This is because the Q-learning computes all the Q-values at each episode (the Q-table contains 27962 states), and the neural network training requires processing too.

Table 4.3 shows the percentage of each correct action for both LSMT and Q-Learning. We used one-year data to calculate these percentages. For the LSTM, we used the predicted action. While For Q-learning, we used the last obtained Q-table.

TABLE 4.2: Comparison between LSTM and Q-Learning execution Time

Algorithm \ Iterations	5	10	50	100
LSTM	4.18s	4.96s	10.74s	17.16s
Q-learning	48mn	2h15mn	28h07mn	75h.11mn

We started by determining the percentage accuracy for one day for each action (Use Battery, Buy from the Grid, Store in the Battery and use Solar Production). Then we calculated the average percentage for the entire test period. It appears that LSTM provides better results than Q-Learning. The worst rate for LSTM is 80.59% of correct predicted actions (Action using the Battery), while the best for Q-Learning is 69.8% of chosen actions (Action buying from the Grid).

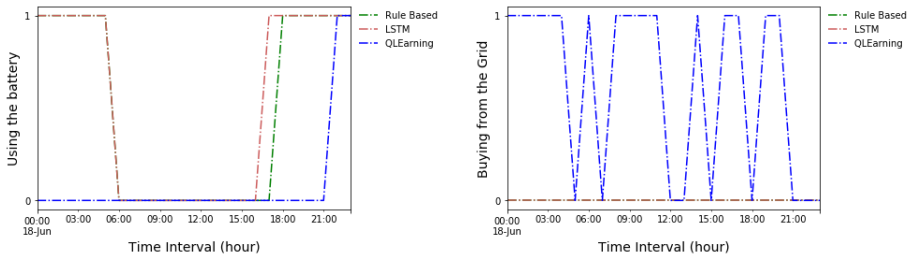
TABLE 4.3: Accuracy of daily predicted actions LSTM vs Q-Learning

Percentage \ Algorithm	LSTM	QLearning
Action Using Battery	80.59%	39.27%
Action Buying from the Grid	81.08%	69.8%
Action Storing in the Battery	95.36%	57.55%
Action Using the solar Production	95.35%	57.06%

In this section, the study will be done over one random weekday during one year. It is 18 June 2019. We compared the chosen action for each proposed approach in figures 4.10 and 4.11.

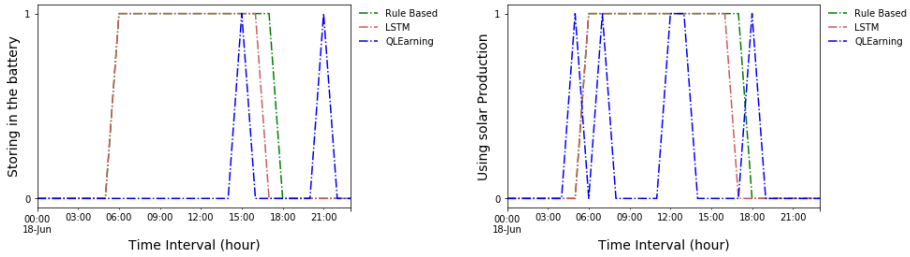
In figure 4.10a, we are showing the predicted use of Battery action. The rule-based results are in green, the LSTM is in Indian red, and the Q-Learning is in blue. The light brown color shows when the action performed by the Rule-Based and LSTM match precisely. For the Q-Learning, the results match from 06:00 to 16:00. We can observe the three different colors when there is a time shift in action decisions. 4.10b shows the Predicted action Buy from the grid with LSTM vs. Q-learning vs. RB.

Figure 4.11 presents the results of the predicted actions for storing in the Battery and using the solar energy produced. 4.11a shows the Predicted action Store in the Battery obtained with RB, LSTM, and Q-learning in green, Indian red, and blue, respectively. While 4.11b shows the predicted action. Use Solar Production with LSTM, Rule-Based, and Q-Learning. Each time the value is equal to one, we will apply an action. For example, at 12:00, both LSTM and RB predicted that we would store in the Battery and use the solar production since we have an extra production compared to the consumption.



(A) Predicted Use of Battery Action LSTM vs Qlearning vs RB (B) Predicted Buy from the grid Action LSTM vs Qlearning vs RB

FIGURE 4.10: DL vs RL predicted action for one day: Predicted Use of Battery and Predicted Buy from the grid



(A) Predicted Store in the Battery Action LSTM vs Qlearning vs RB (B) Predicted Use Solar Production Action LSTM vs Qlearning vs RB

FIGURE 4.11: DL vs RL predicted action for one day: Predicted Store and Predicted use of solar energy

LSTM results The last figures go from 4.12 to 4.15. It concerns the reinforcement part. We respectively compare predicted actions with real optimal actions from the test dataset for a) Use battery action, b) grid usage, c) battery storage action, and d) solar usage. The orchid color shows when the actions are exactly matching. The blue color represents the real optimal action, and the coral color depicts the predicted action. We can observe the blue and coral colors when there is a small-time shift in action decisions.

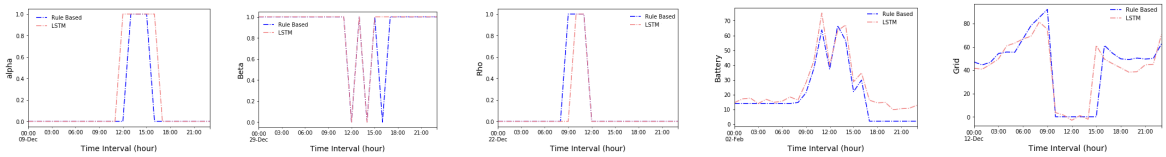
Tab. 4.5 to 4.8 give respective error metrics for the learning phase. But the most important result is summarized in Tab. 4.7. We can see the mean daily accuracy of the different actions in all use cases. The exact resolution chooses the correct action for a specific day at a well-defined time. For each use case, we compared the action resulting from the prediction algorithm to the action provided by the exact resolution. We then made a daily average over the entire data set. We consider that these results are excellent since they globally vary between 80 and 95 % except for only one action (*theta*). Furthermore, the Q-learning algorithm is slower than the LSTM. Tab. 4.2 shows the execution time for both LSTM and Q-learning.

TABLE 4.4: Accuracy of daily predicted actions

Use Case / %	Case 1	Case 2	Case 3	Case 4
percentage Alpha	90.51%	80.59%	80.51%	80.23%
percentage Beta	90.46%	81.08%	80.93%	79.99%
percentage Gamma	-	95.36%	95.40%	95.11%
percentage Rho	95.34%	95.35%	95.39%	93.58%
percentage Teta	-	-	66.8%	89.39%

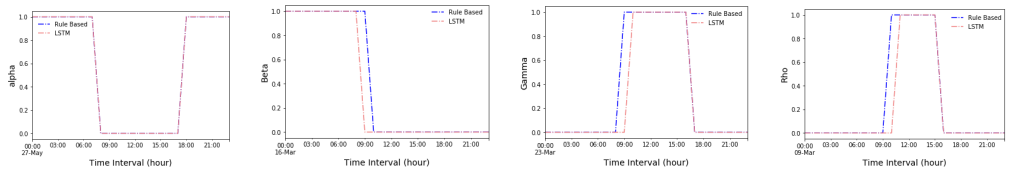
TABLE 4.5: KPI performance for First case LSTM resolution

KPI / Action	MSE	RMSE	R^2
Alpha	0.053	0.23	0.37
Beta	0.054	0.23	0.77
Rho	0.005	0.07	0.98



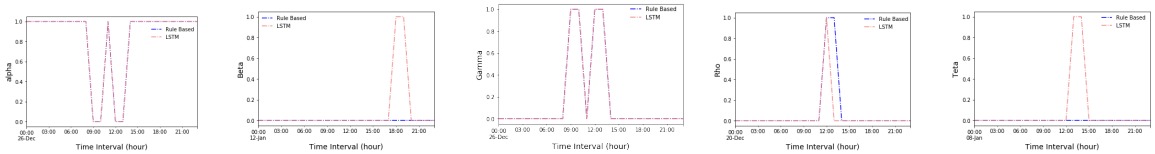
(A) Predicted Use Battery Action Prediction vs Real (B) Predicted Buy Action vs Real (C) Predicted Use Solar Action Prediction vs Real (D) Predicted Battery SoC vs Real (E) Predicted amount of energy bought from the grid vs Real

FIGURE 4.12: First case LSTM resolution



(A) Predicted Use Battery Action vs Real (B) Predicted Buy Action vs Real (C) Predicted Store Action vs Real (D) Predicted Use Solar Action vs Real

FIGURE 4.13: Second case LSTM resolution



(A) Predicted Use battery Action vs Real (B) Predicted Buy Action vs Real (C) Predicted Store Action vs Real (D) Predicted Use solar Action vs Real (E) Predicted Satisfaction D_t Action vs Real

FIGURE 4.14: Third case LSTM resolution

TABLE 4.6: KPI performance for Second case LSTM resolution

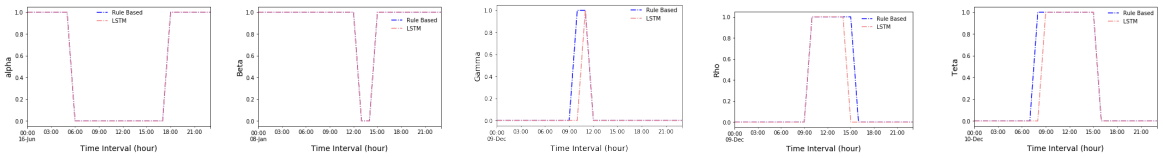
Action \ KPI	MSE	RMSE	R^2
Alpha	0.16	0.40	0.25
Beta	0.16	0.39	-6.08
Gamma	0.005	0.07	0.976
Rho	0.005	0.07	0.976

TABLE 4.7: KPI performance for Third case LSTM resolution

Action \ KPI	MSE	RMSE	R^2
Alpha	0.16	0.40	0.25
Beta	0.15	0.39	-6.08
Gamma	0.004	0.066	0.978
Rho	0.004	0.067	0.977
Teta	0.29	0.53	0.0

TABLE 4.8: KPI performance for Fourth case LSTM resolution

Action \ KPI	MSE	RMSE	R^2
Alpha	0.167	0.40	0.285
Beta	0.17	0.41	0.24
Gamma	0.007	0.08	0.965
Rho	0.022	0.14	0.903
Teta	0.064	0.25	0.729



(A) Predicted Use battery Action vs Real (B) Predicted Buy Action vs Real (C) Predicted Store Action vs Real (D) Predicted Use solar Action vs Real (E) Predicted Satisfaction D_t Action vs Real

FIGURE 4.15: Fourth case LSTM resolution

4.3 Cellular network Use case

The management of the increase in Internet of things (IoT) network devices and applications [132],[133] requires massive optimization algorithm processing and high resource capacities. On the other hand, a serious will to reduce the carbon footprint of IT in general pushes network operators to find alternatives to massive computing tools in order to replace them with more nature-friendly approaches. Machine learning (ML) algorithms, as we are using through reinforcement, constitute

a general method that, when well trained, efficiently replaces such massive model evaluators and network simulators.

So, instead of simulating the network behaviors for each new configuration or optimizing the parameters for each new user profile, a reinforcement learning algorithm will learn the model and provide results for any further configuration without requiring repetitive evaluations. The trained ML is hence considered a good model approximator.

Of course, any machine learning approach will require an initial training process that consumes energy and CPU, and this is one main interest in our work. But training is done once, and model execution is then straightforward with minimal energy consumption.

The architecture of cellular networks demands high levels of optimization to prevent network anomalies. Out-of-range communications, obstacles, unusual human presence, etc., typically cause anomalies. Because this is the most important issue that can pose service degradation and directly affect the different network functionalities, it is a central problem we target to solve in this work.

In the same context, the unmanned aerial vehicles (UAV) [134, 135] have achieved a high development in different fields, especially in networking offloading [136] by the appearance of interesting architectures [1] and solutions for both navigation and communications that can be integrated with mobile networks to provide alternative solutions in case of disconnection or network problems.

In this section, we propose to use UAVs on an on-demand basis for offloading base stations to provide services to connected devices. We first propose an optimal model that aims to maximize the data collection by taking into account the UAV battery. Then we define a deep reinforcement learning environment based on two techniques: Q-learning and LSTM. They learn the optimal model. After that, we present a comparison study for both proposed approaches based on real cellular network datasets, where we analyze the exactness and convergence times.

4.3.1 Related work

In this section, we present the relevant studies in the area of UAV-assisted new network generations.

Pakrooh *et al.* [137] highlight UAV services for mobile devices and IoT systems. The UAVs serve IoT communication, processing of generated data, etc. However, the UAV navigation mechanism is not studied in this work.

In [138], the author introduced an architecture for orchestrating and managing 5G and beyond networks that operate over a heterogeneous

[1] European project 5G! Drone

infrastructure with UAVs' aid. The approach introduces UAVs as a service for mission-critical with challenging 5G connectivity. They collect and process heterogeneous data from different sources while improving network connectivity. However, the overall architecture needed to include the trajectory planning of UAVs, which is necessary for real platforms.

Demir *et al.* [139] proposed UAV deployment in V2X communication field. They tackle the energy-efficient deployment problem of UAVs as mobile RSUs while considering vehicle users latency ad back-haul link capacity constraints and assuring efficient power control of vehicle users.

In [140], authors introduced Software Defined Networking (SDN) as the control plane that manages UAVs, legacy vehicles, and MEC servers. Then, they proposed an SDN-enabled UAV-assisted computation offloading optimization framework for vehicular networks to minimize the system vehicle computing tasks cost. Indeed, UAVs are deployed to reduce the MEC server's load and minimize vehicle-to-MEC network latency. The safety applications to clarify the practical interest of the proposed solution is not studied in this work.

4.4 Modeling and ILP optimization of the Quality of Service

This section presents a simple network architecture and the ILP (Integer Linear Programming) optimization model. It can be easily complexified with more details that will be discussed at the end of the section [102].

4.4.1 Network Model and architecture

We consider the (CDRs) Call Detail Records published as a part of the Big Data Challenge launched by Telecom Italia in 2014.

The architecture of the proposed model is presented in figure 4.16. We consider a centralized cellular infrastructure composed of a coordinator base station and base stations (BS). Generally, mobile users' activities are different at a particular time and cell. When detecting an anomaly (presented in chapter 3) [141], and [142]), each BS communicates the predicted anomalous cell ID(s) with the coordinator to initiate remedial actions. The coordinator launched Unmanned Aerial Vehicles (UAVs) with the mission of collecting data in overloaded cells. We concentrate on drone actions. We define four actions: send a drone to the damaged cell, commit offloading, return the drone to the charging station, and idle.

The main block is based on the ILP optimization model and will be replaced by the reinforcement learning approach. The inputs to the model are the CDR data, the usual load of each terrestrial base station, and the characteristics of the drones to be deployed. The output is the actions on drones: deploy and eventually backhauling. We use the terms backhauling or offloading with the same meaning.

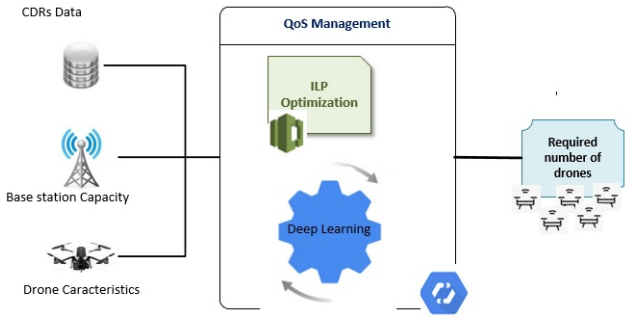


FIGURE 4.16: QoS Management architecture

4.4.2 ILP Optimization

Below, we formulate the ILP [143] optimization problem governing the studied use case where we aim to afford adequate QoS to applications. In Table 4.9, we describe the different variables used to define the model that we aspire to solve.

TABLE 4.9: Notations per cell

Notation	*Description*
$C_y(t)$	Capacity of the base station y during slot t (maximum amount of data that can be served with the BS in the time slot)
$A(t)$	The user total demands in the case of anomaly
$B(t)$	The user demand that need to be served with the drone: Backhauling
$D_x(t)$	The battery capacity of the drone x at time slot t
$\overline{D}$	The maximum charge of the drone battery.
$\underline{D}$	Minimum State-of-charge (SoC) of the drone.
$E_b(t)$	The energy consumed by the drone when performing backhauling action at time slot t
$E_s(t)$	The energy consumed by the drone when performing send action at time slot t
α_t	Binary, sending the Drone
β_t	Binary, making the backhauling action
γ_t	Binary, return action
ρ_t	return action
κ_t	Binary, Drone is charging (state)
Cmax	Int, number of Cells
Nmax	Int, Max number of drones

We try to serve the bandwidth demand first with cellular capacities. We have in our possession information concerning the users' demand per hour. If more than the BS capacity is needed, that means that we are facing an anomaly. In this case, we have to deploy a drone to support the areas in need during the required number of time slots. If no handover [144] is possible and the drone is not sent, the client will suffer a bad QoS and instability [145].

The examined use case is a simplistic outlier emergency coverage scenario without any distinction of traffic types. When an anomaly occurs, it is due to an unexpected traffic profile. The process is done for 24 hours based on outlier prediction.

The objective function proposed here is maximizing the backhauling and hence minimizing the dropped data.

$$Maximize : bDrone{=}\sum_{t=1}^{T} \beta{\times}(A(t){-}C(t))$$

subject to:

$$\alpha_t + \beta_t + \gamma_t + \rho_t = 1 \tag{4.25}$$

$$\alpha_t * \beta_t = 0\ ,\ \alpha_t * \gamma_t = 0\ ,\ \alpha_t * \rho_t = 0\ ,\ \beta_t * \gamma_t = 0\ ,\ \beta_t * \rho_t = 0\ ,\ \gamma_t * \rho_t = 0 \tag{4.26}$$

$$D(t) = D(t-1) - \beta_t \times E_b(t) - \alpha_t \times E_s(t) - \gamma_t \times E_s(t) \tag{4.27}$$

$$if(\gamma_t)\ then\ (\kappa_{t+1} == 1) \tag{4.28}$$

$$if(\kappa_t + anomaly_{t+1} \geq 2)\ then\ (\alpha_{t+1} == 1) \tag{4.29}$$

$$if(anomaly_t - \kappa_t) == 1\ then\ (\beta_t == 1) \tag{4.30}$$

$$if(anomaly_t == 0)\ then\ \rho_t == 1 \tag{4.31}$$

$$if(\kappa_{t+1} == 0)\ then\ (\gamma_t == 1) \tag{4.32}$$

Eqs.4.25 and 4.26 ensure that at a time step t, only one action could be applied. Eq. 4.27 allows the update of the state of charge of the drone battery depending on the actions carried out. Eq. 4.28 means that κ is 1 (return to charging station) in case the drone is doing a return operation. Eq. 4.29 indicates that α is performed if there is a predicted anomaly. Eq. 4.30 indicates when we can make backhauling. 4.31 tells when there is an idle state, and 4.32 triggers the return action. This code is run with CPLEX (academic version) and is downloadable on the 4th author's webpage.

4.5 Reinforcement Learning Approach

ILP gives an optimal solution for problems if we correctly represent the case of the study. However, it is not without a non-trivial computation cost and time. Also, it has to be run anytime a minimal change happens to the dataset. This engenders a waste of energy and resources. The idea we present here is to implement an environment describing our use case and make a machine learning system learn the actions to perform for such a problem. The method used is called reinforcement learning. It is a branch of machine learning, and the algorithms we will use are called Q-Learning and deep reinforcement.

Many advantages will emanate from this approach, like much less consumed energy and CPU resources and building a quite simple autonomous

optimization tool that can be embedded in small devices. Another significant result will be the possibility of adapting the NN to changing conditions in the system. The main idea behind reinforcement learning is that it is represented as an agent that will interact with the environment. The overall result looks like a 'black box system' that behaves like the required solver.

States identify the environment. In this work, our goal is to design a learning agent able to optimally control the drone to maximize the backhauling in case of an outlier. The agent will take action and, according to the result, will receive some reward. We defined the various elements needed by our reinforcement learning algorithm.

- **State:** The identified state space is composed of four variables. The first one is the time of the day. The second one is if there is an anomaly or not. The third one is the state of charge of the drone, and the last one is to indicate if the drone is in the charging station or is deployed. The time and anomaly are inputs. However, the battery level of the drone and the drone location is calculated based on the chosen action.
- **Action:** The chosen action defines the following states. The action space considered in this context is:
  - *Send:* The agent will send the drone to assist the BS.
  - *Backhauling:* The drone is already deployed and will serve the over-demand.
  - *Charge:* The agent will send the drone back to the charging station.
  - *Idle:* The agent will not make any order to the drone. The drone will stay in stationary mode.
- **Reward:** When performing the correct action, the agent will receive a reward. It is defined as optimizing drone usage while maximizing cellular coverage. A binary value presents the reward. After each action, the reward will be calculated.

Fig. 4.17 shows the finite automate with the different states and actions. The reward isn't present in this figure since we assume that the agent is taking the optimal action and that the reward is maximal here. For one day and one hour, granularity if the agent makes only optimal actions, the reward will equal 24.

If we consider $t = 2$, our state is as follows [2,1,80,0], which means that there is an anomaly, the drone battery is equal to 80%, and the drone is deployed. The optimal action, in this case, is backhauling.

Fig.4.18 describes the internal update of the state according to the chosen action. The first check is if there is an anomaly, then we need to

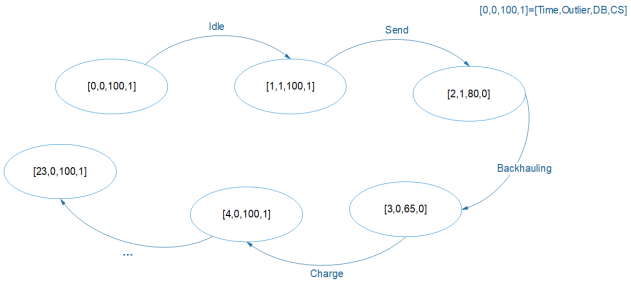


FIGURE 4.17: Finite automate for one day

verify whether there is enough battery in the drone and if it is already deployed. Since we are dealing with only one base station, in this case, the drone location is important information to apply the right action. Once we apply adequate action, we need to calculate the updated drone capacity and location. In our studied use case, we assumed that when applying the send action, we need to update the battery drone by subtracting 20% of the battery State of Charge (SoC). And while we apply the backhauling action, we deduct 15% of the battery SoC.

The reinforcement learning algorithm, specifically the Q-Learning, is applied to the defined environment. The first step is the exploration phase. First, a Q-Learning table is initialized randomly. It presents the states and actions. The algorithm defines the optimal policy that assigns a probability distribution to each state on the set of actions. The estimated probability is called Q-Value. Through the training, the Q-Learning will explore the various possible actions and update the Q values based on the received reward after each action. Then it will learn the Q value of the optimal policy. This step is called the exploitation phase. Finally, we will repeat this treatment for many episodes.

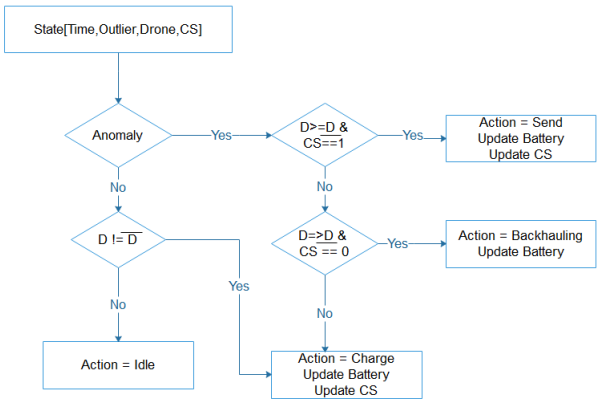


FIGURE 4.18: General flow diagram for the proposed algorithm

4.5.1 Results and discussion

In this section, we present the performance results of the Q-learning approach.

Execution Time

We tested our environment and our algorithm using as input a single day and the entire dataset, which is two months. We subsequently performed several resolutions by varying the number of iterations from 100,500,1000,5000 and 10000. We then calculated the execution time for each iteration. For one day, as seen in Tab. 4.10, the execution time is exponential. It goes from 0.32mn to 668.55 mn for respectively 100 and 10000 iterations. The used server has the following characteristics: 32 Go of RAM, two processors, Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz, with a 15360 KB cache size.

TABLE 4.10: Q-Learning execution Time

Input data / Iterations	One day	Two months
100	0.32mn	668.77mn
500	2.51mn	-
1000	6.37mn	-
5000	152.24mn	-
10000	668.55mn	-

Fig. 4.19 shows the results of the obtained rewards and Penalties for 5000 iterations. The rewards oscillate from 0 to 14, and the penalties oscillate from 10 to 24. For one day, the sum of Penalties and reward is equal to 24 since we are training and testing for 24 hours, and we can apply one action per hour. The highest obtained reward is 14, which is represented by the red point for iterations 263, 3570, 4422, and 4589, respectively.

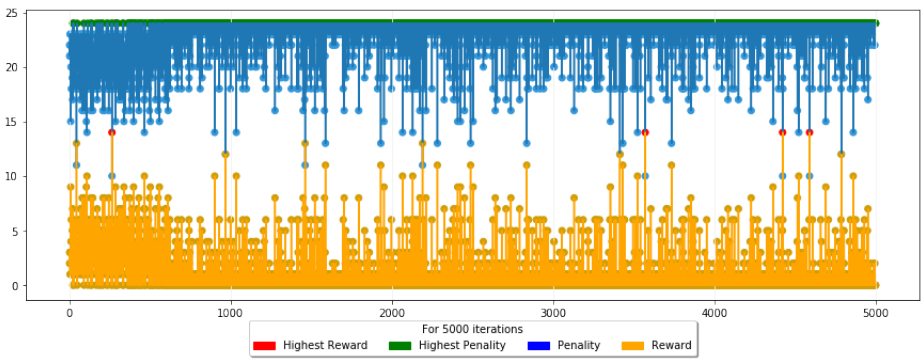


FIGURE 4.19: Reward and Penalties for 5000 iterations

Fig. 4.20 presents a part of the result, and this in order to have an understandable insight. We can observe the results of iterations 8800 to 10000. In this simulation, we are also training and testing for a period of one day. The highest obtained reward in this resolution is equal to 17. The red point for iteration 8861 represents it. The reward oscillates from 1 to 17, and the penalties from 7 and 23.

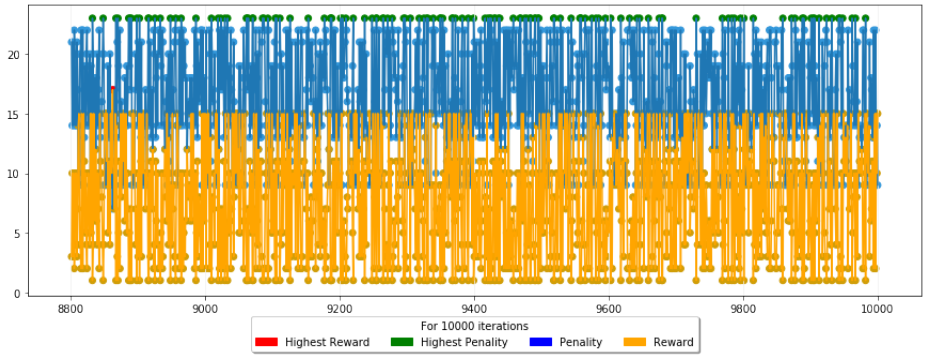


FIGURE 4.20: Reward and Penalties for 5000 iterations

Other results with 100 iterations using the entire data-set correspond to 2 months (November and December). We have 1464 entrees 24×61. Since each entry corresponds to an hour, we can apply action that we can have under optimal circumstances 1664 reward. The highest obtained reward in this resolution is equal to 272. It is received in 6 iterations. The reward oscillates from 167 to 272, and the penalties from 1192 and 1297.

TABLE 4.11: Q-Learning mean values

data	iteration	Rewards	Penalty
One day	1000	10.7%	89.3%
One day	10000	31.45%	68.55%
61 days	100	14.76%	85.24%

4.5.2 From Deep LSTM to Transformers

Q-learning is slow to converge and requires very long periods of training. In general, we think that it should be enriched with more brilliant exploration steps to avoid going into unnecessary states. This is not a good point if we compare it to the exact ILP. The second approach is to learn from the results obtained by CPLEX. The idea is to build a neural network set of weights as fast as possible by learning the minimum number of results from the ILP. Then, we execute the neural network (NN) on new data and see the rewards. Q learning does not need an

evaluated ILP model but needs the development of the environment instead. Deep learning needs something to learn from, so it requires an initial set of good results. So both RL solutions have good points.

Deep LSTM First, our choice goes to a hybrid LSTM network with six cells and 100 hidden layers, completed by a dense layer at the output. Inputs for the NN are anomalies and time taken in the ILP over the training set (2/3 of the two months of CDRs). The training labels correspond to the chosen action.

Since the results were not satisfying, we decided to move to transformers and evaluate them in our use case.

LSTM Transformers LSTM Transformers was initially introduced in [4]. LSTM, Transformer is an architecture for transforming one sequence into another with the help of two parts (Encoder and Decoder). The Encoder is on the left, and the Decoder is on the right see Fig. 4.21. Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times. The model is constituted of three-part [146]:

- **Encoder :** A stack of several recurrent units (LSTM or GRU cells for better performance) where each accepts a single element of the input sequence, collects information for that element, and propagates it forward.
- **Encoder Vector :** This is the final hidden state produced from the encoder part of the model. This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions. It acts as the initial hidden state of the decoder part of the model.
- **Decoder:** A stack of several recurrent units where each predicts an output y_t at a time step t. Each recurrent unit accepts a hidden state from the previous unit and produces an output as well as its own hidden state.

Results The results give an approximate score of 80% of good actions over the small training interval we use (2 months of total data). A Mean Square Error of 0.04 is obtained. In Fig.4.22, we can see that Deep LSTM very finely follows CPLEX actions. Alpha (when the drone is sent) and Beta (backhauling) are shown, and the results are, in our sense, excellent. Figure 4.22 presents the results of predicted actions with LSTM, Q-learning, and CPLEX.

We have compared two RL techniques to enforce a simple process of sending a drone forth and back to offload an anomalous cell. The process that is evaluated is very simple. It could be increased with more

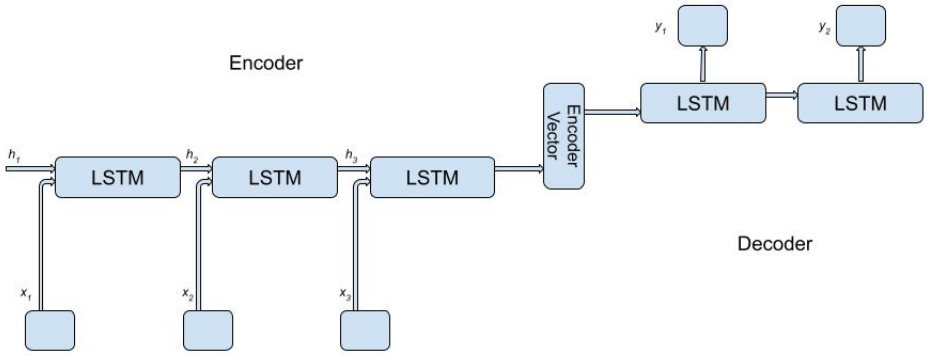
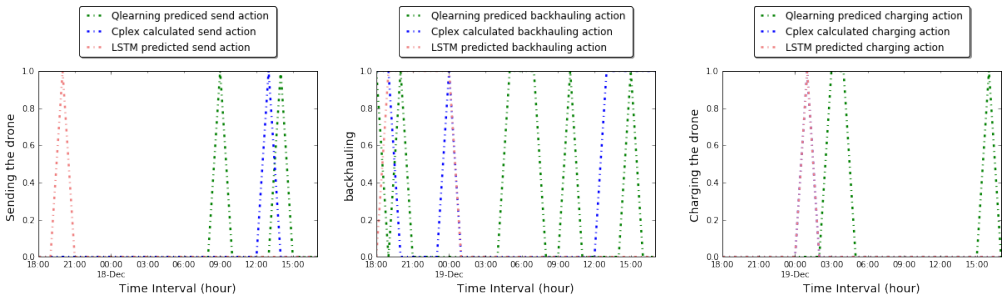


FIGURE 4.21: Encoder-decoder sequence to sequence model



(A) Predicted send drone Action vs Qlearning vs Real (B) Predicted backhauling Action vs Qlearning vs Real (C) Predicted charging drone Action vs Qlearning vs Real

FIGURE 4.22: Third case LSTM resolution

NP-hard conditions, such as a number of anomalous cells that are superior to the number of drones and different quality of service requirements in different cells. However, adding such constraints first renders the ILP problem to solve very large, and it may not be possible to solve at all.

Also, when increased, the Q-learning model requires considerable time to converge as it is a blind trial process exploring all the states of the Q table. The only stable process here is the deep neural network. It learns quite fast, and the MSE converges very fast. It could be improved with a different Sequence to Sequence or transformer LSTM architecture.

4.6 Conclusion

We designed a deep reinforcement learning algorithm to manage two systems (energy and cellular). It differs from the state-of-the-art approaches as the state/action/reward approach is replaced with a recurrent 'on policy' system. It is adapted to several use cases that optimize energy management for buildings that host a microgrid and do not require large datasets or long training times. Three steps are presented to calculate the final neural network weights.

First, we use deep LSTM to obtain the daily consumption and solar production prediction. Second, ILP models or Rule-Based algorithms are solved to calculate optimal actions for the predicted data. Third, we train a second deep LSTM with the optimal samples evaluated by the models. Results show that the reinforcement system makes excellent decisions that reach 95 % exactness compared to the optimal test set and independently from the complexity of the use case.

We used the same methodology for the cellular use case. We started by using DLSTM to obtain the daily load of a base station. These predictions are used to detect if an anomaly occurs. Once an abnormal behavior is detected, we resolve the sending drone problem to support the base station using ILP, Reinforcement, and DLSTM.

The first problem encountered in this work was model representation with ILP. It would be nice to have easier methods or tools that facilitate those formulations. The second problem faced was the long execution time of Q-learning. Finally, identifying the neural network's meta parameters and what relevant information to introduce constituted a challenging task.

Chapter 5

Conclusion and Perspectives

5.1 Conclusions

The prominent matter of the thesis revolves around the analysis of real-time serial datasets and the management and optimization of resources. We investigated five time-series datasets from various resources.

The first concerns energy consumption data from two buildings, The first is a startup incubator, and the second is a student dorm building. The second dataset details the photovoltaic panels' energy production. The third and the fourth datasets consist of CDRs metadata that contains detailed information about the exchanges (call in/out, SMS in/out, and internet usage) with the cellular networks. Orange Senegal provided the first CDRs dataset as part of the D4D challenge dataset. Telecom Italia provided the second dataset as part of the big data challenge. Finally, the last dataset contains traces of taxi mobility in Rome, Italy. It includes GPS traces of trajectories taken by the taxis.

Various information is joined to those data sets. We can cite the temporal information as the first information. The second one is specific to the geographical information of the third, fourth, and fifth datasets. We studied these datasets to propose time series data analysis and exploitation tools. The different extracted information has been used not only to validate the proposed classification and prediction algorithms but also to enhance renewable energy usage according to demand and network resource management.

Since these datasets are massive, we started by applying data analysis techniques to clean the dataset and extract the relevant knowledge. The second application was the use of advanced tools such as Machine learning algorithms.

- We applied in a first-place classification algorithm to two use cases (Cellular and Energy). For the cellular dataset, we have done a spatial classification. In fact, for each base station, according to the

daily traffic, we identified different classes. We then used the 1D-CNN to classify the different base stations and the cell to which they belong. We used an MLP to make a time series classification for the energy use case. We categorized the types of energy consumption into three classes (Critical, Delayable, and comfort). Then the MLP guesses to which type the input belongs.

- In the second place, the problem that we faced and tried to solve is we could predict with high accuracy the future values of the time series data sets? This thesis addresses this issue by providing an in-depth study of various machine learning algorithms (SVR, LSTM, GRU, DLSTM). The results proved the high efficiency of the DLSTM prediction model. We evaluated our model on the five datasets. For the first one, we predicted the energy consumption of the buildings. We used the second dataset to train our model on predicting photovoltaic production. The third and the fourth datasets were used to train the DLSTM to predict the future base station network load. Using the last dataset, we trained our neural network model to predict the taxis' trajectory in Rome.
- Since the classification and the prediction alone do not respond to all our requirements and do not perform well when applied to a large dataset with multivariate predictions. We combine the classification and the prediction algorithms. Our architecture is based on MLP and LSTM for the first two datasets and composed of CNN-1d and LSTM for the second one for the sake of prediction.

In the third chapter, we exploit a more advanced feature that can be found in RNN, and we propose novel applications that can benefit from RNN characteristics.

- The first studied application is semantic compression, where we emphasize the need to compress and only transmit relevant information using a particular network: LORA.
- The second studied application is transfer learning. We showed the efficiency of this method in two cases. The first one deals with the lack of data and the second one is data personalizing.
- The last application described in this chapter is anomaly detection, which is highly dependent on the work carried out in the first chapter. In fact, to detect the anomalies, we compare the actual behavior with a dynamic threshold determined with the predicted normal behavior.

The fourth chapter was dedicated to developing resource Management and optimization Algorithms. Two use cases were studied. The first relates to energy, where we aim to develop an autonomous energy management system. We designed a deep reinforcement learning algorithm to manage energy systems. It differs from the state-of-the-art approaches as the state/action/reward technique is replaced with a recurrent 'on policy' system. It is adapted to several use cases that optimize energy management for buildings that host a microgrid and do not require large datasets or long training times. Three steps are presented to calculate the final neural network weights. First, we use deep LSTM to obtain the daily consumption and solar production prediction. Second, ILP models or Rule-Based algorithms are solved to calculate optimal actions for the predicted data. Third, we present a QLearning model, and finally, we train a second deep LSTM with the optimal samples evaluated by the models.

The second one is the cellular network, where we propose to use UAVs on an on-demand basis for offloading base stations to provide services to connected devices. Hence, the goal is to improve the network's service quality. Then, we present the ILP formulation and two RL models. In the first, a Q-learning technique is used where the agent interacts with an environment and makes decisions considering the same objective function. A real dataset has validated the proposed solution. It helps network operators to efficiently manage their infrastructure and allows them to implement self-organized and autonomous networks that can face a plethora of unexpected data. A second deep learning model, the LSTM transformers, is studied for future use. It is based on deep LSTM networks. It learns faster than Q learning and reaches 80% of good actions.

To conclude this work, we remind you that we investigated various time series datasets to propose a general architecture to deal with TS. The proposed methodology allows us to apply classification and prediction methods to the available datasets. The results helped us exploit new approaches like transfer learning or anomaly detection to integrate them after that into dynamic resource management and optimization algorithm.

5.2 Perspectives

From the work carried out during the Ph.D., several major lines of future work are available:

- Test our proposed solution to other time series datasets

- Embark the LSTM algorithm onto smaller IoT
- Possibility to introduce a non-cyclical component to anomaly detection
- The presented algorithms in the fourth chapter could be integrated with an emulator to benefit from a real battery behavior, and they could be used in a different context: for QoS planning in an edge cloud RAN resource management system.
- Apply other deep Reinforcement techniques such as DQN or policy gradient to compare it with our algorithm and apply these methods to other reinforcement problems such as quality of service enforcement in cellular systems, drone deployment, and edge caching devices.
- Develop a lightweight reinforcement neural network to deploy on LORA micro-controllers.

Bibliography

[1] Steffen Hölldobler, Sibylle Möhle, and Anna Tigunova. "Lessons Learned from AlphaGo." In: *YSIP*. 2017, pp. 92–101.

[2] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[3] Joseph Redmon and Anelia Angelova. "Real-time grasp detection using convolutional neural networks". In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 1316–1322.

[4] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[6] José F Torres et al. "Deep learning for time series forecasting: a survey". In: *Big Data* 9.1 (2021), pp. 3–21.

[7] Jeffrey Andrews et al. "What will 5G be?" In: *Selected Areas in Communications, IEEE Journal on* 32 (May 2014). DOI: 10.1109/JSAC.2014.2328098.

[8] Pouyan Pourbeik, Prabha S Kundur, and Carson W Taylor. "The anatomy of a power grid blackout-root causes and dynamics of recent major blackouts". In: *IEEE Power and Energy Magazine* 4.5 (2006), pp. 22–29.

[9] Jason Lines et al. "Classification of household devices by electricity usage profiles". In: *International conference on intelligent data engineering and automated learning*. Springer. 2011, pp. 403–412.

[10] Robert Thomas Olszewski. *Generalized feature extraction for structural pattern recognition in time-series data*. Carnegie Mellon University, 2001.

[11] Pengyu Hao et al. "Feature selection of time series MODIS data for early crop classification using random forest: A case study in Kansas, USA". In: *Remote Sensing* 7.5 (2015), pp. 5347–5369.

[12] Hassan Ismail Fawaz et al. "Inceptiontime: Finding alexnet for time series classification". In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[13] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 International joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 1578–1585.

[14] F. Karim et al. "LSTM Fully Convolutional Networks for Time Series Classification". In: *IEEE Access* 6 (2018), pp. 1662–1669. DOI: 10.1109/ACCESS.2017.2779939.

[15] Li Wei and Eamonn Keogh. "Semi-supervised time series classification". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 748–753.

[16] Huanhuan Li et al. "Adaptively constrained dynamic time warping for time series classification and clustering". In: *Information Sciences* 534 (2020), pp. 97–116.

[17] Seif Eddine Hammami. "Dynamic network resources optimization based on machine learning and cellular data mining". PhD thesis. Evry, Institut national des télécommunications, 2018.

[18] https://www.sciencedirect.com/topics/computer-science/logistic-regression.

[19] https://datascientest.com/regression-logistique-quest-ce-que-cest.

[20] Aleksandra Bartosik and Hannes Whittingham. "Evaluating safety and toxicity". In: *The Era of Artificial Intelligence, Machine Learning, and Data Science in the Pharmaceutical Industry*. Elsevier, 2021, pp. 119–137.

[21] https://dataanalyticspost.com/Lexique/k-nearest-neighbours/.

[22] Stefan Oehmcke, Oliver Zielinski, and Oliver Kramer. "kNN ensembles with penalized DTW for multivariate time series imputation". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 2774–2781.

[23] Glenn Shafer. "Conditional probability". In: *International Statistical Review/Revue Internationale de Statistique* (1985), pp. 261–275.

[24] https://mrmint.fr/naive-bayes-classifier.

[25] https://medium.com/swlh/decision-trees-classifier-aba3c53e14b9.

[26] Richa Sharma, Aniruddha Ghosh, and PK Joshi. "Decision tree approach for classification of remotely sensed satellite data using open source support". In: *Journal of Earth System Science* 122.5 (2013), pp. 1237–1247.

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

[28] Adele Cutler, D Richard Cutler, and John R Stevens. "Random forests". In: *Ensemble machine learning*. Springer, 2012, pp. 157–175.

[29] Tim Menzies et al. *Sharing data and models in software engineering*. Morgan Kaufmann, 2014.

[30] S Abirami and P Chitra. "Energy-efficient edge based real-time healthcare support system". In: *Advances in Computers*. Vol. 117. 1. Elsevier, 2020, pp. 339–368.

[31] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[32] `https://datascientest.com/convolutional-neural-network`.

[33] Aicha Dridi et al. "An artificial intelligence approach for time series next generation applications". In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE. 2020, pp. 1–6.

[34] Georgia Papacharalampous, Hristos Tyralis, and Demetris Koutsoyiannis. "Predictability of monthly temperature and precipitation using automatic time series forecasting methods". In: *Acta Geophysica* 66.4 (2018), pp. 807–831.

[35] Seif Eddine Hammami et al. "Network planning tool based on network classification and load prediction". In: *2016 IEEE Wireless Communications and Networking Conference*. IEEE. 2016, pp. 1–6.

[36] Yılmaz Akdi, Elif Gölveren, and Yasin Okkaoğlu. "Daily electrical energy consumption: Periodicity, harmonic regression method and forecasting". In: *Energy* 191 (2020), p. 116524.

[37] Koichi Kurumatani. "Time series forecasting of agricultural product prices based on recurrent neural networks and its evaluation method". In: *SN Applied Sciences* 2.8 (2020), pp. 1–17.

[38] J. Kolter and Joseph Ferreira. *A Large-Scale Study on Predicting and Contextualizing Building Energy Usage*. 2011. URL: `https://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3759`.

[39] Claudio Martani et al. "ENERNET: Studying the dynamic relationship between building occupancy and energy consumption". In: *Energy and Buildings* 47 (2012), pp. 584 –591. ISSN: 0378-7788. DOI: `https://doi.org/10.1016/j.enbuild.2011.12.037`. URL: `http://www.sciencedirect.com/science/article/pii/S0378778811006566`.

[40] Shailendra Singh and Abdulsalam Yassine. "Big data mining of energy time series for behavioral analytics and energy consumption forecasting". In: *Energies* 11.2 (2018), p. 452.

[41] Jian Cao, Zhi Li, and Jian Li. "Financial time series forecasting model based on CEEMDAN and LSTM". In: *Physica A: Statistical Mechanics and its Applications* 519 (2019), pp. 127–139.

[42] Ah Chung Tsoi. "Recurrent neural network architectures: an overview". In: *International School on Neural Networks, Initiated by IIASS and EMFCSC*. Springer. 1997, pp. 1–26.

[43] `https://www.ibm.com/cloud/learn/recurrent-neural-networks`.

[44] `https://datascientest.com/recurrent-neural-network`.

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

[46] Zheng Zhao et al. "LSTM network: a deep learning approach for short-term traffic forecast". In: *IET Intelligent Transport Systems* (2017).

[47] `https://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm/`.

[48] Ben Krause et al. "Multiplicative LSTM for sequence modelling". In: *arXiv preprint arXiv:1609.07959* (2016).

[49] Wendong Zheng et al. "Understanding the property of long term memory for the LSTM with attention mechanism". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 2708–2717.

[50] Fardin Syed, Riccardo Di Sipio, and Pekka Sinervo. "Bidirectional Long Short-Term Memory (BLSTM) neural networks for reconstruction of top-quark pair decay kinematics". In: *arXiv preprint arXiv:1909.01144* (2019).

[51] Jin Zhang et al. "Data augmentation and dense-LSTM for human activity recognition using WiFi signal". In: *IEEE Internet of Things Journal* 8.6 (2020), pp. 4628–4641.

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

[53] Antoine Bernard et al. "Embedding ML algorithms onto LP-WAN sensors for compressed communications". In: *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE. 2021, pp. 1539–1545.

[54] *STM32L476 Microcontroller*. `https://www.st.com/en/microcontrollers-microprocessors/stm32l476rg.html`. 2021.

[55] *Semtech's SX1276MB1MAS*. `https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276mb1mas`. 2021.

[56] *STM32 Nucleo Expansion Board*. https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/stm32-nucleo-expansion-boards/x-nucleo-lpm01a.html. 2021.

[57] Jin Wang et al. "Dimensional sentiment analysis using a regional CNN-LSTM model". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2016, pp. 225–230.

[58] Tae-Young Kim and Sung-Bae Cho. "Predicting Residential Energy Consumption using CNN-LSTM Neural Networks". In: *Energy* (2019).

[59] Amin Ullah et al. "Action recognition in video sequences using deep bi-directional LSTM with CNN features". In: *IEEE Access* 6 (2017), pp. 1155–1166.

[60] John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017).

[61] Shu Lih Oh et al. "Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats". In: *Computers in biology and medicine* 102 (2018), pp. 278–287.

[62] https://www.egauge.net/.

[63] https://www.influxdata.com/.

[64] https://bokeh.org/.

[65] S. E. Hammami et al. "Network planning tool based on network classification and load prediction". In: *2016 IEEE Wireless Communications and Networking Conference*. 2016, pp. 1–6. DOI: 10.1109/WCNC.2016.7565166.

[66] Danny Qiu et al. "Classifying Urban Fabrics into Mobile Call Activity with Supervised Machine Learning". In: *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE. 2021, pp. 1948–1953.

[67] Telecom Italia. *Telecom Italia Big Data Challenge. (2015):* https://http://www.telecomitalia.com/tit/en/bigdatachallenge.html. (Visited on 11/20/2016).

[68] Orange. *D4D Challenge:* https://http://www.d4d.orange.com. (Visited on 04/10/2017).

[69] Aicha Dridi et al. "STAD: Spatio-temporal anomaly detection mechanism for mobile network management". In: *IEEE Transactions on Network and Service Management* 18.1 (2020), pp. 894–906.

[70] Seifeddine Hammami. "Dynamic network resources optimization based on machine learning and cellular data mining". In: 2018. URL: http://www.theses.fr/2018TELE0015.

[71] Aicha Dridi et al. "Transfer learning for classification and prediction of time series for next generation networks". In: *ICC 2021-IEEE International Conference on Communications*. IEEE. 2021, pp. 1–6.

[72] Mohammed Laroui et al. "Energy management for electric vehicles in smart cities: a deep learning approach". In: *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2019, pp. 2080–2085.

[73] Bryan E Usevitch. "A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000". In: *IEEE signal processing magazine* 18.5 (2001), pp. 22–35.

[74] V. M. Suresh et al. "Powering the IoT through embedded machine learning and LoRa". In: *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. 2018, pp. 349–354. DOI: 10.1109/WF-IoT.2018.8355177.

[75] Jiheon Kang and Doo-Seop Eom. "Offloading and Transmission Strategies for IoT Edge Devices and Networks". In: *Sensors* 19 (Feb. 2019), p. 835. DOI: 10.3390/s19040835.

[76] K. MATSUDA and M. KUBOTA. "Compound Compression Method for Gathering Traffic of IoT/CPS Data". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 2019. DOI: 10.1109/WF-IoT.2019.8767326.

[77] Gimenez Petrov. "Internet-Draft SCHC-over-LoRaWAN". In: *Internet Engineering Task Force*. 2019.

[78] Aicha Dridi et al. "Deep learning semantic compression: Iot support over lora use case". In: *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*. IEEE. 2019, pp. 1–6.

[79] Bastien Mainaud, Vincent Gauthier, and Hossam Afifi. "Cooperative communication for wireless sensors network: a mac protocol solution". In: *2008 1st IFIP Wireless Days*. IEEE. 2008, pp. 1–5.

[80] S. J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[81] H. Ismail Fawaz et al. "Transfer learning for time series classification". In: *2018 IEEE International Conference on Big Data (Big Data)*. 2018, pp. 1367–1376. DOI: 10.1109/BigData.2018.8621990.

[82] S. E. Hammami et al. "Network planning tool based on network classification and load prediction". In: *2016 IEEE Wireless Communications and Networking Conference*. 2016, pp. 1–6. DOI: 10.1109/WCNC.2016.7565166.

[83] Muazzam Maqsood et al. "Transfer learning assisted classification and detection of Alzheimer's disease stages using 3D MRI scans". In: *Sensors* 19.11 (2019), p. 2645.

[84] Avi Bleiweiss. "LSTM Neural Networks for Transfer Learning in Online Moderation of Abuse Context". In: 2018. URL: http://www.insticc.org.

[85] Chao Lu et al. "Transfer Learning for Driver Model Adaptation in Lane-Changing Scenarios Using Manifold Alignment". In: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[86] Md Abul Bashar, Richi Nayak, and Nicolas Suzor. "Regularising LSTM classifier by transfer learning for detecting misogynistic tweets with small training set". In: *Knowledge and Information Systems* 62.10 (2020), pp. 4029–4054.

[87] Ahmed Soua and Hossam Afifi. "Adaptive data collection protocol using reinforcement learning for VANETs". In: *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE. 2013, pp. 1040–1045.

[88] Ioannis D Apostolopoulos and Tzani A Mpesiana. "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks". In: *Physical and Engineering Sciences in Medicine* (2020), p. 1.

[89] Cherifa Boucetta et al. "Adaptive Range-based Anomaly Detection in Drone-assisted Cellular Networks". In: *IWCMC*. IEEE. 2019, pp. 1239–1244.

[90] Samira Rezaei et al. "Automatic fault detection and diagnosis in cellular networks using operations support systems data". In: *IEEE/IFIP NOMS*. 2016.

[91] Jun Wu et al. "CellPAD: Detecting performance anomalies in cellular networks via regression analysis". In: *2018 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE. 2018, pp. 1–9.

[92] Seif Eddine Hammami, Hassine Moungla, and Hossam Afifi. "Proactive Anomaly Detection Model for eHealth-Enabled Data in Next Generation Cellular Networks". In: *IEEE ICC*. 2018.

[93] Mohammad Gharbieh et al. "Spatiotemporal stochastic modeling of IoT enabled cellular networks: Scalability and stability analysis". In: *IEEE Trans. Commun.* (2017).

[94] Bilal Hussain, Du Qinghe, and Ren Pinyi. "Deep Learning-Based Big Data-Assisted Anomaly Detection in Cellular Networks". In: Dec. 2018.

[95] Bilal Hussain et al. "Artificial Intelligence-powered Mobile Edge Computing-based Anomaly Detection in Cellular Networks". In: *IEEE Transactions on Industrial Informatics* (Nov. 2019), pp. 1551–3203. DOI: 10.1109/TII.2019.2953201.

[96] Y. Hu, M. Chen, and W. Saad. "Joint Access and Backhaul Resource Management in Satellite-Drone Networks: A Competitive Market Approach". In: *IEEE Transactions on Wireless Communications* (2020), pp. 1–1.

[97] Fatima Bousbaa et al. "GeoUAVs: A new geocast routing protocol for fleet of UAVs". In: *Computer Communications* 149 (Oct. 2019), pp. 259–269. DOI: 10.1016/j.comcom.2019.10.026.

[98] Cherifa Boucetta et al. "Optimizing drone deployment for cellular communication coverage during crowded events". In: *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*. IEEE. 2019, pp. 622–627.

[99] Chérifa Boucetta et al. "Heuristic Optimization Algorithms for QoS Management in UAV Assisted Cellular Networks". In: *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE. 2020, pp. 01–06.

[100] Aicha Dridi et al. "A Novel Deep Reinforcement Approach for IIoT Microgrid Energy Management Systems". In: *IEEE Transactions on Green Communications and Networking* 6.1 (2021), pp. 148–159.

[101] Aicha Dridi et al. "Deep Recurrent Learning versus Q-Learning for Energy Management Systems in Next Generation Network". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2021, pp. 1–6.

[102] Aicha Dridi et al. "Reinforcement Learning Vs ILP Optimization in IoT support of Drone assisted Cellular Networks". In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 4589–4594.

[103] Aysun Aslan, Gülce Bal, and Cenk Toker. "Dynamic Resource Management in Next Generation Networks based on Deep Q Learning". In: *2020 28th Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2020, pp. 1–4.

[104] Paulo Lissa et al. "Deep reinforcement learning for home energy management system control". In: *Energy and AI* 3 (2021), p. 100043.

[105] Fayiz Alfaverh, Mouloud Denai, and Yichuang Sun. "Demand response strategy based on reinforcement learning and fuzzy reasoning for home energy management". In: *IEEE Access* 8 (2020), pp. 39310–39321.

[106] Tanguy Levent et al. "Energy Management for Microgrids: a Reinforcement Learning Approach". In: *ISGT-Europe 2019 - IEEE PES Innovative Smart Grid Technologies Europe*. Bucharest, France: IEEE, Sept. 2019, pp. 1–5. DOI: 10.1109/ISGTEurope.2019.8905538. URL: https://hal.archives-ouvertes.fr/hal-02382232.

[107] Hepeng Li, Zhiqiang Wan, and Haibo He. "A Deep Reinforcement Learning Based Approach for Home Energy Management System". In: *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*. IEEE. 2020, pp. 1–5.

[108] R. Sutton. "Reinforcement Learning: An Introduction - Stanford University". In: ().

[109] Liang Yu et al. "Deep reinforcement learning for smart home energy management". In: *IEEE Internet of Things Journal* 7.4 (2019), pp. 2751–2762.

[110] José R Vázquez-Canteli and Zoltán Nagy. "Reinforcement learning for demand response: A review of algorithms and modeling techniques". In: *Applied energy* 235 (2019), pp. 1072–1089.

[111] Mohit Sewak. "Deep Q Network (DQN), Double DQN, and Dueling DQN". In: 2019.

[112] Matin Farhoumandi, Quan Zhou, and Mohammad Shahidehpour. "A review of machine learning applications in IoT-integrated modern power systems". In: *The Electricity Journal* 34.1 (2021), p. 106879.

[113] "IEC61970: Energy management system application program interface (EMS-API). IEC; 2005." In: ().

[114] A. G. Tsikalakis and N. D. Hatziargyriou. "Centralized control for optimizing microgrids operation". In: *2011 IEEE Power and Energy Society General Meeting*. 2011, pp. 1–8. DOI: 10.1109/PES.2011.6039737.

[115] Luu Ngoc An and Tran Quoc-Tuan. "Optimal energy management for grid connected microgrid by using dynamic programming method". In: *2015 IEEE Power Energy Society General Meeting*. 2015, pp. 1–5. DOI: 10.1109/PESGM.2015.7286094.

[116] Muhammad Fahad Zia, Elhoussin Elbouchikhi, and Mohamed Benbouzid. "Microgrids energy management systems: A critical review on methods, solutions, and prospects". In: *Applied energy* 222 (2018), pp. 1033–1055.

[117] Yuankun Liu, Dongxia Zhang, and Hoay Beng Gooi. "Optimization strategy based on deep reinforcement learning for home energy management". In: *CSEE Journal of Power and Energy Systems* 6.3 (2020), pp. 572–582.

[118] Bin Xu et al. "Parametric study on reinforcement learning optimized energy management strategy for a hybrid electric vehicle". In: *Applied Energy* 259 (2020), p. 114200.

[119] Esmat Samadi, Ali Badri, and Reza Ebrahimpour. "Decentralized multi-agent based energy management of microgrid using reinforcement learning". In: *International Journal of Electrical Power & Energy Systems* 122 (2020), p. 106211.

[120] Guodong Du et al. "Deep reinforcement learning based energy management for a hybrid electric vehicle". In: *Energy* 201 (2020), p. 117591.

[121] Daeil Lee, Awwal Mohammed Arigi, and Jonghyun Kim. "Algorithm for Autonomous Power-Increase Operation Using Deep Reinforcement Learning and a Rule-Based System". In: *IEEE Access* 8 (2020), pp. 196727–196746. DOI: 10.1109/ACCESS.2020.3034218.

[122] Elena Mocanu et al. "On-line building energy optimization using deep reinforcement learning". In: *IEEE transactions on smart grid* 10.4 (2018), pp. 3698–3708.

[123] I. Brahmia et al. "Robust Data Predictive Control Framework for Smart Multi-Microgrid Energy Dispatch Considering Electricity Market Uncertainty". In: *IEEE Access* 9 (2021), pp. 32390–32404. DOI: 10.1109/ACCESS.2021.3060315.

[124] Aicha Dridi et al. "Machine Learning Application to Priority Scheduling in Smart Microgrids". In: *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE. 2020, pp. 1695–1700.

[125] Joseph El Hayek. "Le problème de bin-packing en deux-dimensions, le cas non-orienté: résolution approchée et bornes inférieures." PhD thesis. Université de Technologie de Compiègne, 2006.

[126] Shi Sha et al. "A thermal-balanced variable-sized-bin-packing approach for energy efficient multi-core real-time scheduling". In: *Proceedings of the on Great Lakes Symposium on VLSI 2017*. 2017, pp. 257–262.

[127] Dennis Binu and BR Rajakumar. *Artificial Intelligence in Data Mining: Theories and Applications*. Academic Press, 2021.

[128] https://www.tricentis.com/artificial-intelligence-software-testing/ai-approaches-rule-based-testing-vs-learning/.

[129] Mohammed Laroui et al. "SO-VMEC: Service offloading in virtual mobile edge computing using deep reinforcement learning". In: *Transactions on Emerging Telecommunications Technologies* (), e4211.

[130] Richard S Sutton and Andrew G. Barto. "Reinforcement Learning: An introduction". In: London, England, 2015, pp. 1–398.

[131] Haşim Sak, Andrew Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *Fifteenth annual conference of the international speech communication association*. 2014.

[132] Ala Al-Fuqaha et al. "Internet of things: A survey on enabling technologies, protocols, and applications". In: *IEEE communications surveys & tutorials* 17.4 (2015), pp. 2347–2376.

[133] Moungla B.Nour. "Internet of Things Mobility Over Information-Centric/Named-Data Networking". In: *in IEEE Internet Computing*, vol. 24, no. 1, pp. 14–24, 1 Jan.–Feb. 2020, doi: 10.1109/MIC.2019.2963187.

[134] Mohammed Laroui et al. "Autonomous UAV Aided Vehicular Edge Computing for Service Offering". In: *IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6.

[135] Rohit Chaurasia and Vandana Mohindru. "Unmanned Aerial Vehicle (UAV): A Comprehensive Survey". In: *Unmanned Aerial Vehicles for Internet of Things (IoT) Concepts, Techniques, and Applications* (2021), pp. 1–27.

[136] Nader S Labib et al. "The Rise of Drones in Internet of Things: A Survey on the Evolution, Prospects and Challenges of Unmanned Aerial Vehicles". In: *IEEE Access* 9 (2021), pp. 115466–115487.

[137] Rambod Pakrooh and Ali Bohlooli. "A Survey on Unmanned Aerial Vehicles-Assisted Internet of Things: A Service-Oriented Classification". In: *Wireless Personal Communications* (2021), pp. 1–35.

[138] Cristiano Bonato Both et al. "System Intelligence for UAV-Based Mission Critical with Challenging 5G/B5G Connectivity". In: *arXiv e-prints* (2021), arXiv–2102.

[139] Uygar Demir, Cenk Toker, and Özgür Ekici. "Energy-efficient deployment of uav in v2x network considering latency and backhaul issues". In: *IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. 2020, pp. 1–6.

[140] Liang Zhao et al. "A Novel Cost Optimization Strategy for SDN-Enabled UAV-Assisted Vehicular Computation Offloading". In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[141] Cherifa Boucetta et al. "Adaptive Range-based Anomaly Detection in Drone-assisted Cellular Networks". In: (2019), pp. 1239–1244. DOI: 10.1109/IWCMC.2019.8766446.

[142] Chérifa Boucetta et al. "Heuristic Optimization Algorithms for QoS Management in UAV Assisted Cellular Networks". In: (2020), pp. 01–06. DOI: 10.1109/GLOBECOM42002.2020.9322243.

[143] Hatem Ibn-Khedher et al. "OPAC: An optimal placement algorithm for virtual CDN". In: *Computer Networks* 120 (2017), pp. 12–27. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2017.04.009. URL: https://www.sciencedirect.com/science/article/pii/S1389128617301391.

[144] Kaouthar SETHOM and Hossam AFIFI. "Requirements and adaptation solutions for transparent handover between wifi and bluetooth". In: *ICC 2004 (2004 IEEE International Conference on Communications)*. 2004.

[145] Mohammed Laroui et al. “Driving path stability in VANETs”. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 1–6.

[146] `https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346`.

ECOLE
DOCTORALE

Titre : Une nouvelle approche d'apprentissage en profondeur efficace pour le traitement des séries Temporelles utilisant la classification, la prédiction et le renforcement : Cas d'utilisations Energie et télécommunications

Mots clés : Séries temporelles, Apprentissage machine, Réseaux de neurones récurrents, Apprentissage profond, Apprentissage par renforcement, Détection d'anomalie

Résumé : La croissance massive des capteurs (température, humidité, accéléromètre, capteur de position) et des appareils mobiles (smartphones, tablettes, smartwatch ...) fait que la quantité de données générées augmente de manière explosive. Cette immense quantité de données peut être collectée et gérée. Le travail réalisé durant cette thèse vise à proposer en un premier temps une approche qui traite un type de données spécifique qui sont les séries temporelles. Pour ce faire nous avons utilisé des méthodes de classification basées sur des réseaux de neurones convolutifs ainsi que des multi layer perceptron afin d'extraire les informations pertinentes. Nous avons par la suite eu recours à l'utilisation des réseaux de neurones récurrents pour réaliser les prédictions. Les données utilisées provenaient de plusieurs sources : Données de consommation énergétique, données de production d'énergies renouvelables, données cellulaires, données de trace GPS de taxi. Nous avons également investigué plusieurs autres méthodes telles que la compression sémantique ainsi que le transfer learning. Les deux méthodes décrites précédemment nous permettent pour la première de ne transmettre que les poids des réseaux de neurones ou en cas d'anomalie détectée d'envoyer les données la constituant. Le transfer learning nous permet quant à lui de réaliser de bonnes prédictions même si les données traitées souffrent d'un manque ou d'un bruit. Ces traitements nous ont permis par la suite de mettre en place des mécanismes dynamiques de détection d'anomalie. L'objectif du dernier volet de la thèse est le développement et l'implémentation d'une solution de management des ressources ayant comme entrée le résultat des phases précédentes. Pour mettre en place cette solution de gestion des ressources nous avons utilisé plusieurs approches tel que l'apprentissage par renforcement, la résolution exacte ou encore des réseaux de neurones récurrents. Une première application est la mise en place d'un système de management de l'énergie et la seconde est la gestion du déploiement des drones pour assister les réseaux cellulaires en cas d'anomalies.

Title : A novel efficient time series deep Learning Approach using Classification, Prediction and reinforcement: Energy and Telecom use case

Keywords : Time series, Machine learning, Recurrent neural network, Deep learning, Reinforcement learning, Anomaly detection

Abstract : The massive growth of sensors (temperature, humidity, accelerometer, position sensor) and mobile devices (smartphones, tablets, smartwatches) increases the amount of data generated explosively. This immense amount of data can be collected and managed. The work carried out during this thesis aims first to propose an approach that deals with a specific type of data, which are time series. First, we used classification methods based on convolutional neural networks and multilayer perceptrons to extract the relevant information. We then used recurrent neural networks to make the predictions. We treated several time series data: energy, cellular, and GPS taxi track data. We also investigated several other methods like as semantic compression and transfer learning. The two described methods above allow us for the first to transmit only the weight of the neural networks, or if an anomaly is detected, send the anomalous data. Transfer learning allows us to make good predictions even if the data is missing or noisy. These methods allowed us to set up dynamic anomaly detection mechanisms. The objective of the last part of the thesis is to develop and implement a resource management solution having as input the result of the previous phases. We used several methods to implement this resource management solution, such as reinforcement learning, exact resolution, or recurrent neural networks. The first application is the implementation of an energy management system. The second application is the management of the deployment of drones to assist cellular networks when an anomaly occurs.

Institut Polytechnique de Paris
91120 Palaiseau, France